ionic

目錄

Ionic 中文文档

Ionic CSS 文档

> 来源：Ionic CSS 文档

头部/Header

Headers are fixed regions at the top of a screen that can contain a title label, and left/right buttons for navigation or to carry out various actions.

Headers come in a variety of default color options:

bar-light

```
<divclass="bar bar-header bar-light"><h1class="title">bar-light</h1></div>
```

bar-stable

```
<divclass="bar bar-header bar-stable"><h1class="title">bar-stable</h1></div>
```

bar-positive

```
<divclass="bar bar-header bar-positive"><h1class="title">bar-positive</h1></div>
```

bar-calm

```
<divclass="bar bar-header bar-calm"><h1class="title">bar-calm</h1></div>
```

bar-balanced

```
<divclass="bar bar-header bar-balanced"><h1class="title">bar-balanced</h1></div>
```

bar-energized

```
<divclass="bar bar-header bar-energized"><h1class="title">bar-energized</h1></div>
```

bar-assertive

```
<divclass="bar bar-header bar-assertive"><h1class="title">bar-assertive</h1></div>
```

bar-royal

```
<divclass="bar bar-header bar-royal"><h1class="title">bar-royal</h1></div>
```

bar-dark

```
<divclass="bar bar-header bar-dark"><h1class="title">bar-dark</h1></div>
```

ionic二级导航栏/Sub Header

A secondary header bar can be placed below the original header bar. There are quite a few more ways to customize Headers. Have a look at Button Bars to get other ideas on how it could be used.

```
<divclass="bar bar-header"><h1class="title">Header</h1></div><divclass="bar bar-subheader
```

Also, remember to include the `has-subheader` CSS class to your `ion-content` directive.

ionic内容/Content

The content area in Ionic is the scrollable viewport of your app. While your headers and footers will be fixed to the top and bottom, respectively, the content area will fill the remaining available space.

* For more content options, see the Content docs.

ionic底部/Footer

Footers are regions at the bottom of a screen that can contain various types of content.

```
<divclass="bar bar-footer bar-balanced"><divclass="title">Footer</div></div>
```

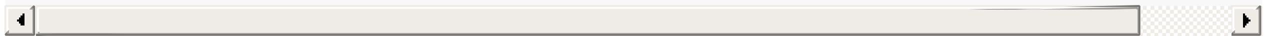
Footers have the same color options as the headers, just use `bar-footer` instead of `bar-header`. If a title is present in the footer, any buttons will automatically be placed on the correct side of the title in relation to how the markup was written, such as:

```
<divclass="bar bar-footer"><buttonclass="button button-clear">Left</button><divclass="tit
```



Additionally, if no title is present and a right side button is required, you'll need to add `pull-right` to the right side button, such as:

```
<divclass="bar bar-footer"><buttonclass="button button-clear pull-right">Right</button></
```



ionic按钮/Buttons

Ah, the Button, an essential part of any mobile experience. Like the Header, they come in the full spectrum of Ionic's default colors.

By default a button has `display: inline-block` applied. Other options include `block` buttons for a full width.

```
<buttonclass="button">
  Default
</button><buttonclass="button button-light">
  button-light
</button><buttonclass="button button-stable">
  button-stable
</button><buttonclass="button button-positive">
  button-positive
</button><buttonclass="button button-calm">
  button-calm
</button><buttonclass="button button-balanced">
  button-balanced
</button><buttonclass="button button-energized">
  button-energized
</button><buttonclass="button button-assertive">
  button-assertive
</button><buttonclass="button button-royal">
  button-royal
</button><buttonclass="button button-dark">
  button-dark
</button>
```

Block Buttons

Adding `button-block` to a button applies `display: block` display. A block button will however go 100% of its parent's width. In the example, the button's containing content element also has `padding` applied, so there is some breathing room between the edge of the device and the buttons.

```
<buttonclass="button button-block button-positive">
  Block Button
</button>
```

Full Width Block Buttons

Adding `button-full` to a button not only applies `display: block`, but also removes borders on the left and right, and any border-radius which may be applied. This style is useful when the button should stretch across the entire width of the display. Additionally, the button's parent element *does not* have `padding` applied.

```
<buttonclass="button button-full button-positive">
  Full Width Block Button
</button>
```

Different Sizes

Adding `button-large` to a button makes it larger, adding `button-small` makes it smaller.

```
<buttonclass="button button-small button-assertive">
  Small Button
</button><buttonclass="button button-large button-positive">
  Large Button
</button>
```

Outlined Buttons

Use `button-outline` to apply an outline button style, which also has a transparent background.

Note: The text and border will take the color of the applied button style, meaning `button-positive` will result in blue text and border, with a transparent background.

```
<buttonclass="button button-outline button-positive">
  Outlined Button
</button>
```

Clear Buttons

Add `button-clear` to remove the border and make the text stand out.

Note: The text will take the color of the applied button style, meaning `button-positive` will result in blue text and no border instead of a blue background.

```
<buttonclass="button button-clear button-positive">
  Clear Button
</button>
```

Icon Buttons

Icons can easily be added to any button by using either the built in Ionicons, or any custom font-pack you choose. Learn more about icons.

Icons can also be set with a child element inside the button, however, assigning the icon directly to the button reduces the number of elements in the DOM.

```
<buttonclass="button"><iclass="icon ion-loading-c"></i> Loading...
</button><buttonclass="button icon-left ion-home">Home</button><buttonclass="button icon-
```



Buttons in Headers

When buttons are placed in headers or footers, they take the style of the bar by default, so you don't have to use the extra style classes. To change this, add the desired style class.

```
<divclass="bar bar-header"><buttonclass="button icon ion-navicon"></button><h1class="titl
```

Clear Buttons in Headers

For a more minimal approach to header buttons, simply add the `button-clear` classname to remove the background button color and border.

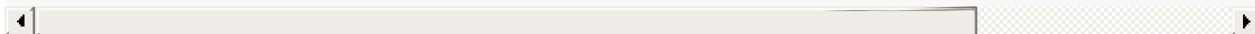
```
<divclass="bar bar-header"><buttonclass="button button-icon icon ion-navicon"></button><d
```



Button Bar

Buttons can also be easily grouped together using the `button-bar` classname. In this example, a button bar was added to the default header bar, as well as in the main content area of the screen.

```
<divclass="button-bar"><aclass="button">First</a><aclass="button">Second</a><aclass="butt
```



列表/List

The List is a common and simple way of displaying... that's right, a list. This is a widely used interface across most current mobile OS's, and can include content ranging from basic text all the way to buttons, toggles, icons, and thumbnails.

The list view is a very versatile and powerful component. List views support various interaction modes such as editing, swipe to edit, drag to reorder, and pull to refresh.

For more power you can use Ionic's AngularJS directives. Check out the AngularJS list docs to get more detailed information.

```
<ulclass="list"><liclass="item">
      ...
    </li></ul>
```

* For a more extensive overview on Lists, read AngularJS List Docs

List Dividers

List items can also be dividers to organize and group the list items. Use the `item-divider` class to create a divider for any child element of the list. By default list item dividers will have a different background color and font-weight, but this is easily customizable.

```
<divclass="list"><divclass="item item-divider">
    Candy Bars
  </div><aclass="item"href="#">
    Butterfinger
  </a>  ...

</div>
```

List Icons

Lists can have icons assigned either to the left and/or right side of each list item, and the alignment classes should be assigned to each `item` element. Icons can easily be added to any item by using either the built in Ionicons, or any custom font-pack you choose. Learn more about icons.

Use `item-icon-left` to line up the icon to the left, and `item-icon-right` to set the icon to the right. When a list item has an icon on both sides then both classes will need to be applied.

This example uses an `<a>` element for each item, which allows the entire list item to be tappable. If the item is an `<a>` or `<button>` element, and no icon has been added to the right, then a small right arrow will automatically be added.

In the example, the first item only has a left aligned icon. The second item has both left and right side icons. The third item has no a right side icon assigned (whichs defaults to the right arrow). Additionally, the third item also adds an `item-note`. The fourth icon has added a `badge` element.

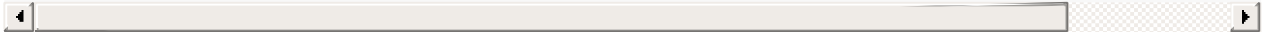
```
<divclass="list"><aclass="item item-icon-left"href="#"><iclass="icon ion-email"></i>
    Check mail
  </a><aclass="item item-icon-left item-icon-right"href="#"><iclass="icon ion-chatbubble-
    Call Ma
    <iclass="icon ion-ios-telephone-outline"></i></a><aclass="item item-icon-left"href="#
    Record album
    <spanclass="item-note">
      Grammy
    </span></a><aclass="item item-icon-left"href="#"><iclass="icon ion-person-stalker"></
    Friends
    <spanclass="badge badge-assertive">0</span></a></div>
```

List Buttons

Use `item-button-right` or `item-button-left` to place a button within an item.

```
<divclass="list"><divclass="item item-button-right">
    Call Ma
    <buttonclass="button button-positive"><iclass="icon ion-ios-telephone"></i></button><

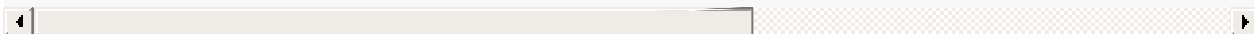
</div>
```



Item Avatars

Item avatars are essentially a showcase of an image larger than an icon, but smaller than a thumbnail. To create a avatar item, use the `item-avatar` classname.

```
<divclass="list"><aclass="item item-avatar"href="#"><imgsrc="venkman.jpg"><h2>Venkman</h2
</div>
```

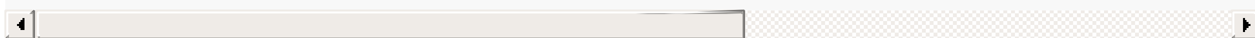


Item Thumbnails

Item Thumbnails are essentially a showcase of an image larger than an icon, and will often span/define the entire height of the list item. To create a thumbnail styled item, use the `item-thumbnail-left` to have it align on the left, and `item-thumbnail-right` for the right side.

```
<divclass="list"><aclass="item item-thumbnail-left"href="#"><imgsrc="cover.jpg"><h2>Prett

</div>
```



Inset Lists

Lists can also be inset inside their container, instead of going full width. The main difference is that a `list list-inset` element has margin. An inset list is similar to a card, except an inset list does not have a box-shadow. Since `list list-inset` does not have a box-shadow, it'll be more performant when scrolling. Inset forms shows other examples of its usage.

```
<divclass="list list-inset"><divclass="item">
      Raiders of the Lost Ark
    </div>    ...

</div>
```

Cards

Cards have become widely used in recent years. They're are a great way to contain and organize information, while also setting up predictable expectations for the user. With so much content to display at once, and often so little screen realestate, cards have fast become the design pattern of choice for many companies, including the likes of Google, Twitter, and Spotify..

For mobile experiences, Cards make it easy to display the same information visually across many different screen sizes. They allow for more control, are flexible, and can even be animated. Cards are usually placed on top of one another, but they can also be used like a "page" and swiped between, left and right.

```
<divclass="card"><divclass="item item-text-wrap">
    This is a basic Card which contains an item that has wrapping text.
  </div></div>
```

Card Headers and Footers

Cards can be customized similarly to how you would fill a normal screen. For example, a card can easily have Headers and Footers placed inside of them. Add the `item-divider` classname above or below the content within the `card` element.

```
<divclass="card"><divclass="item item-divider">
    I'm a Header in a Card!
  </div><divclass="item item-text-wrap">
    This is a basic Card with some text.
  </div><divclass="item item-divider">
    I'm a Footer in a Card!
  </div></div>
```

Card Lists

Use the `list card` classname to create a card with lists.

```
<divclass="list card"><ahref="#"class="item item-icon-left"><iclass="icon ion-home"></i>
    Enter home address
  </a><ahref="#"class="item item-icon-left"><iclass="icon ion-ios-telephone"></i>
    Enter phone number
  </a><ahref="#"class="item item-icon-left"><iclass="icon ion-wifi"></i>
    Enter wireless password
  </a><ahref="#"class="item item-icon-left"><iclass="icon ion-card"></i>
    Enter card information
  </a></div>
```

Card Images

Images look great in cards, and can be combined with lists and other elements.

```
<divclass="list card"><divclass="item item-avatar"><imgsrc="avatar.jpg"><h2>Pretty Hate M
    Start listening
  </a></div>
```



Card Showcase

Here is a showcase of a card using several different items. It begins with the `list card` element, utilizing the `item-avatar` list item, an `item-body` element for images and text, and a footer with the `item-divider` classname.

```
<divclass="list card"><divclass="item item-avatar"><imgsrc="mcfly.jpg"><h2>Marty McFly</h
      This is a "Facebook" styled Card. The header is created from a Thumbnail List item,
      the content is from a card-body consisting of an image and paragraph text. The foot
      consists of tabs, icons aligned left, within the card-footer.
    </p><p><ahref="#"class="subdued">1 Like</a><ahref="#"class="subdued">5 Comments</a></
      Like
    </a><aclass="tab-item"href="#"><iclass="icon ion-chatbox"></i>
      Comment
    </a><aclass="tab-item"href="#"><iclass="icon ion-share"></i>
      Share
    </a></div></div>
```



表单/Forms & Inputs

A `list` is also used to group related input elements. Both `item-input` and `item` is then used to designate each individual input field and it's associated label.

Ionic prefers to create the `item-input` out of the `<label>` element so that when any part of the row is tapped then the underlying input receives focus.

Additionally, developers should be aware of the various HTML5 Input types so users will be presented with the appropriate virtual keyboard.

Text Input: Placeholder Labels

In the example, it'll default to 100% width (no borders on the left and right), and uses the `placeholder` attribute to simulate the input's label. Then the user begins to enter text into the input the placeholder label will be hidden. Notice how `<textarea>` can also be used as a multi-line text input.

```
<divclass="list"><labelclass="item item-input"><inputtype="text"placeholder="First Name">
```



Text Input: Inline Labels

Use `input-label` to place a label to the left of the input element. When the user enters text the label does not hide. Note that there's nothing stopping you from also using a `placeholder` label too.

```
<divclass="list"><labelclass="item item-input"><spanclass="input-label">Username</span><i
```



Text Input: Stacked Labels

Stacked labels always places the label on top of the input. Each item should have `item-stacked-label` assigned, and the input's label should have `input-label` assigned. This example also uses the `placeholder` attribute so users have a hint of what type of text the input is looking for.

```
<divclass="list"><labelclass="item item-input item-stacked-label"><spanclass="input-label
```



Text Input: Floating Labels

Floating labels are just like Stacked Labels, except that their labels animate, or "float" up when text is entered in the input. Each item should have `item-floating-label` assigned, and the input's label should have `input-label` assigned.

Enter text in the example to the right to see the floating labels in action. This example also uses the `placeholder` attribute so user's have a hint of what type of text the input is looking for.

```
<divclass="list"><labelclass="item item-input item-floating-label"><spanclass="input-labe
```



Inset Forms

By default each input item will fill 100% of the width of its parent element (the list). However, you can inset the list using either the `list list-inset` or `card` classnames. The card classname applies a lower box shadow while list-inset does not. Additionally, if the list's parent element has `padding` assigned then this will also give the form an inset appearance.

```
<divclass="list list-inset"><labelclass="item item-input"><inputtype="text"placeholder="F
```



Inset Inputs

Using `list-inset` will inset the entire list, whereas placing `item-input-inset` will inset an input into an individual list item. Placing a button inside the item

```
<divclass="list"><divclass="item item-input-inset"><labelclass="item-input-wrapper"><inpu
      Submit
    </button></div></div>
```



Input Icons

Icons can be easily added to the left of an `item-input` input. Simply add an icon before the `<input>`. By default the icon will take the color of label text. However, you can also use add `placeholder-icon` to give it a placeholder color.

```
<divclass="list list-inset"><labelclass="item item-input"><iclass="icon ion-search placeh
```



Header Inputs

Inputs can also be placed in headers, along with buttons to submit or cancel the form.

```
<divclass="bar bar-header item-input-inset"><labelclass="item-input-wrapper"><iclass="ico
    Cancel
  </button></div>
```

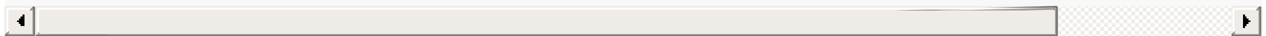


Toggle

A toggle technically is the same thing as an HTML checkbox input, except it looks different and is easier to use on a touch device. Ionic prefers to wrap the checkbox input with the `<label>` in order to make the entire toggle easy to tap or drag.

Toggles can also have colors assigned to them, such as `toggle-assertive` to assign the assertive color.

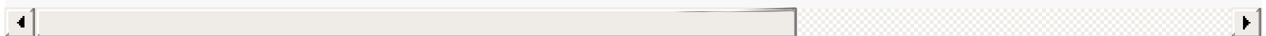
```
<labelclass="toggle"><inputtype="checkbox"><divclass="track"><divclass="handle"></div></d
```



This is an example of multiple toggles within a list. Note the `item-toggle` class was added along side `item` for each item.

```
<ulclass="list"><liclass="item item-toggle">
     HTML5
     <labelclass="toggle toggle-assertive"><inputtype="checkbox"><divclass="track"><divcl

</ul>
```



Checkbox

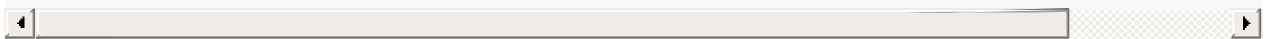
A checkbox is no different than the HTML checkbox input, except it's styled differently. This is an example of multiple checkboxes within a list. Note the `item-checkbox` class was added along side `item` for each item.

Ionic prefers to use the `<label>` element for a checkbox item in order to make the entire checkbox tappable.

Checkboxes can also have colors assigned to them, such as `checkbox-assertive` to assign the assertive color.

```
<ulclass="list"><liclass="item item-checkbox"><labelclass="checkbox"><inputtype="checkbox
     Flux Capacitor
  </li>  ...

</ul>
```



Radio Button List

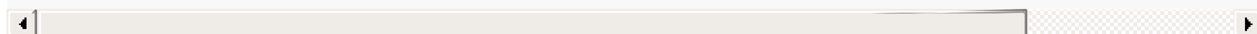
Radio buttons act no differently as standard radio input elements. Following this convention will display a list of radio buttons similarly seen in modern app.

Each `item-radio` must have a radio input with the same input `name` attribute. The `radio-icon` class is used to designate when to show and hide the icon element.

Ionic prefers to use the `<label>` element for a radio item in order to make the entire area tappable.

```
<divclass="list"><labelclass="item item-radio"><inputtype="radio"name="group"><divclass="
      Go
    </div><iclass="radio-icon ion-checkmark"></i></label>  ...

</div>
```



Range

This is a Range. Ranges can be themed to any default Ionic color, and used in various other elements such as a list item or card.

```
<divclass="item range"><iclass="icon ion-volume-low"></i><inputtype="range"name="volume">
```

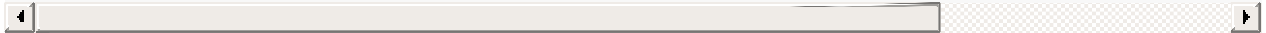


Select

Ionic's select is styled so its appearance is prettied up relative to the browser's default style. However, when the select elements is opened, the default behavior on how to select one of the options is still managed by the browser.

Each platform's user-interface will be different as the user is selecting an option. For example, on a desktop browser you'll see the traditional drop down interface, whereas Android often has a radio-button list popup, and iOS has a custom scroller covering the bottom half of the window.

```
<divclass="list"><labelclass="item item-input item-select"><divclass="input-label">
      Lightsaber
    </div><select><option>Blue</option><optionselected>Green</option><option>Red</option>
```



切换标签/Tabs

Tabs are a horizontal region of buttons or links that allow for a consistent navigation experience between screens. It can contain any combination of text and icons, and is a popular method for enabling mobile navigation.

* For building tabbed interfaces, see the Tabs documentation.

The containing element should have the `tabs` classname, and each tab should have the `tab-item` classname. By default, tabs will be without an icon and text-only.

```
<divclass="tabs"><aclass="tab-item">
    Home
  </a><aclass="tab-item">
    Favorites
  </a><aclass="tab-item">
    Settings
  </a></div>
```

Tabs can be styled to match the standard Ionic colors (the example is using the `default` color. Use these classes to change the color of the tab bar:
`tabs-default``tabs-light``tabs-stable``tabs-positive``tabs-calm``tabs-balanced``tabs-energ`

To hide the tabbar but still show the content, add the `tabs-item-hide` class. Also, whenever you are using tabs, remember to add the `has-tabs` CSS class to .

Icon-only Tabs

Add `tabs-icon-only` along with the `tabs` classname.

```
<divclass="tabs tabs-icon-only"><aclass="tab-item"><iclass="icon ion-home"></i></a><aclas
```

Top Icon Tabs

Classic tabs. Add `tabs-icon-top` along with the `tabs` classname.

```
<divclass="tabs tabs-icon-top"><aclass="tab-item"><iclass="icon ion-home"></i>
    Home
  </a><aclass="tab-item"><iclass="icon ion-star"></i>
    Favorites
  </a><aclass="tab-item"><iclass="icon ion-gear-a"></i>
    Settings
  </a></div>
```

Left Icon Tabs

Add `tabs-icon-left` along with the `tabs` classname.

```
<divclass="tabs tabs-icon-left"><aclass="tab-item"><iclass="icon ion-home"></i>
    Home
  </a><aclass="tab-item"><iclass="icon ion-star"></i>
    Favorites
  </a><aclass="tab-item"><iclass="icon ion-gear-a"></i>
    Settings
  </a></div>
```

Striped Style Tabs

Add `tabs-striped` to an element above the `tabs` classname for Android style tabs.

Optionally, also add `tabs-top` to position the tab at the top

Get granular color control for striped tabs with the `tabs-background-{color}` and `tabs-color-{color}` classes, where `{color}` is any of the ionic color swatches: `default`, `light`, `stable`, `positive`, `calm`, `balanced`, `energized`, `assertive`, `royal`, or `dark`

Note, that to have the header blend with the top tabs, add the `has-tabs-top` class to the header.

```
<divclass="tabs-striped tabs-top tabs-background-positive tabs-color-light"><divclass="ta
        Test
      </a><aclass="tab-item"href="#"><iclass="icon ion-star"></i>
        Favorites
      </a><aclass="tab-item"href="#"><iclass="icon ion-gear-a"></i>
        Settings
      </a></div></div><divclass="tabs-striped tabs-color-assertive"><divclass="tabs"><acl
        Test
      </a><aclass="tab-item"href="#"><iclass="icon ion-star"></i>
        Favorites
      </a><aclass="tab-item"href="#"><iclass="icon ion-gear-a"></i>
        Settings
      </a></div></div>
```



Grid

Ionic's grid system is different than most because of its use of the CSS Flexible Box Layout Module standard. The advantage here is that the devices that Ionic supports, all support flexbox.

Simply add columns you want in a row, and they'll evenly take up the available space. If you want three columns, add three columns, if you want five columns, add five columns. There's no restriction to a 12 column grid, or having to explicitly state how large each column should be. And to add to the crazy, you can vertically align content within each column.

Use the `row` classname is used to designate, surprise, a row, and `col` is used for a column. In the demo to the right we chose to have four, then two, columns, but we could have just as easily used 3, 6, 7, 23, etc., it doesn't matter. Point is, add the number of columns your layout requires and don't worry about figuring out the percentages because it figures it out automagically.

Note: The borders and gray background in the demo were added so it's easier to see the structure.

网格布局/Grid: Evenly Spaced Columns

By default every `col` added inside a `row` will automatically receive an equal amount of the available area. Notice in the code below that no sizes are specified anywhere in the classnames, yet each of the five columns in this example will each evenly take up 20% of the available width (thank you flexbox).

```
<divclass="row"><divclass="col">.col</div><divclass="col">.col</div><divclass="col">.col<
```

Note: The borders and gray background in the demo were added so it's easier to see the structure.

Grid: Explicit Column Sizes

You *can* explicitly state the size of a column if for example you'd want specific columns to be larger than the others in the same row. By default each column will evenly take up the available area, but in the case where a column should be a certain size, Ionic's grid uses a percent system (in contrast to a locked in 12 column grid).

An advantage with this grid system is that you only have to state the percentage for the column that needs it, and the others will still evenly divide up the available areas.

```
<divclass="row"><divclass="col col-50">.col.col-50</div><divclass="col">.col</div><divcla
```

Note: The borders and gray background in the demo were added so it's easier to see the structure.

Explicit Column Percentage Classnames	
.col-10	10%
.col-20	20%
.col-25	25%
.col-33	33.3333%
.col-50	50%
.col-67	66.6666%
.col-75	75%
.col-80	80%
.col-90	90%

Grid: Offset Columns

Columns can also be offset from the left. We'll let the code and demo speak for itself.

```
<divclass="row"><divclass="col col-33 col-offset-33">.col</div><divclass="col">.col</div>
```

Note: The borders and gray background in the demo were added so it's easier to see the structure.

Offset Column Percentage Classnames	
.col-offset-10	10%
.col-offset-20	20%
.col-offset-25	25%
.col-offset-33	33.3333%
.col-offset-50	50%
.col-offset-67	66.6666%
.col-offset-75	75%
.col-offset-80	80%
.col-offset-90	90%

Grid: Vertically Align Columns

Another trick up flexbox's sleeve is the ability to easily vertically align columns. Vertical alignment includes top, center and bottom, and can be applied to every column in a row, or to specific columns.

In the demo, we've made the last column in each row to have the tallest content in order to demonstrate how the content of the others vertically align. The first row shows the default which is to take the same height as the tallest column in the same row.

```
<divclass="row"><divclass="col">.col</div><divclass="col">.col</div><divclass="col">.col<
```

Note: The borders and gray background in the demo were added so it's easier to see the structure.

Responsive Grid

There may be cases where a row of columns will not fit nicely in the available area. The responsive classes can be used to turn each column in a row into its own row at certain breakpoints.

For example, if you want a row of columns to turn in to stacked rows when the viewport is pretty small, you would use the `.responsive-sm` class. The example to the right is a simulation of what it'd look like.

```
<divclass="row responsive-sm"><divclass="col">.col</div><divclass="col">.col</div><divcla
```

Note: The borders and gray background in the demo were added so it's easier to see the structure.

Responsive Class	Break when roughly
.responsive-sm	Smaller than landscape phone
.responsive-md	Smaller than portrait tablet
.responsive-lg	Smaller than landscape tablet

For further configuration, each class uses a Sass variable that can be changed to your liking. There's also a `responsive-grid-break` mixin you can use to create your own classes.

Utility

Ionic comes with a handful of utility classes to help quickly style your design. Each are optional.

Colors

Ionic comes with a set of colors to start with, but as a general rule **colors are meant to be overridden**. We prefer saying that Ionic provides a recommended naming convention for your colors, swatches, themes, etc.

Utility colors are added to help set a naming convention. You'll notice Ionic purposely does not use words like "red" or "blue", but instead have colors which represent an emotion or generic theme.

Let's be realistic, assigning colors is one of the easier tasks in CSS, and each app will have different requirements for colors. Ionic's goal is to provide a clean system to build on top of and maintain, and stays away from dictating how each app chooses to color its custom design.

To customize the colors you can simply override those coming from the `ionic.css` CSS file. Additionally, since Ionic is built using Sass, for more power and flexibility you could also modify and extend the color variables within the `_variables.scss` file.

* For more flexibility, you can Customize Ionic With Sass

Icons

Ionic also comes with its own free and open-sourced icon font, Ionicons, with over 500 icons to choose from.

Simply add `icon` and the Ionicon classname for the icon to show, which can be easily looked up on the Ionicons homepage.

```
<iclass="icon ion-star"></i>
```

While it's possible for buttons to use a child `<i>` to set the icon, they can also set their icon just by setting the buttons own class. Please take a look at button icon docs for more info.

Note: Ionic is certainly not restricted to using only the Ionicons icon pack, so please feel free to use any icons you wish.

Padding

Many components in Ionic purposely have both padding and margin reset set to zero. In many instances apps will have components bleed to the edge of the screen, and by starting each component at zero developers can easily control padding and margins throughout the app.

The `padding` utility classes can be reused to give any element's content some breathing room, meaning it adds a default `10px` between the outer box of the element and its inner content. The following classes are not required for any element, but may be helpful with your layout.

- `padding` Adds padding around every side.
- `padding-vertical` Adds padding to the top and bottom.
- `padding-horizontal` Adds padding to the left and right.
- `padding-top` Adds padding to the top.
- `padding-right` Adds padding to the right.
- `padding-bottom` Adds padding to the bottom.
- `padding-left` Adds padding to the left.

Ionic AngularJS 扩展

> 来源：Ionic AngularJS 扩展

Ionic既是一个CSS框架也是一个Javascript UI库。许多组件需要Javascript才能产生神奇的效果，尽管通常组件不需要编码，通过框架扩展可以很容易地使用，比如我们的AngularIonic扩展。

Ionic遵循视图控制模式，通俗的理解和 Cocoa 触摸框架相似。在视图控制模式中，我们将界面的不同部分分为子视图或包含其他视图的子视图控制器。然后视图控制器"驱动"内部视图来提供交互和UI功能。一个很好的例子就是标签栏（Tab Bar）视图控制器处理点击标签栏在一系列可视化面板间切换。

浏览我们的API文档来了解视图控制器和Ionic中可用的Javascript实用工具。

Ionic 是目前最有潜力的一款 HTML5 手机应用开发框架。通过 SASS 构建应用程序，它 提供了很多 UI 组件来帮助开发者开发强大的应用。 它使用 JavaScript MVVM 框架和 AngularJS 来增强应用。提供数据的双向绑定，使用它成为 Web 和移动开发者的共同选择。

快速学习ionic也可以观看ionic免费视频教程Ionic教程

Headers/Footers（页眉页脚）

ion-header-bar

在内容顶部添加一个固定header栏。

如果应用'bar-subheader'类，就可以成为一个subheader（在下面）。 查看header CSS文档。

用法

```
<ion-header-bar align-title="left" class="bar-positive">
  <div class="buttons">
    <button class="button" ng-click="doSomething()">左侧按钮</button>
  </div>
  <h1 class="title">Title!</h1>
  <div class="buttons">
    <button class="button">右侧按钮</button>
  </div>
</ion-header-bar>
<ion-content>
  一些内容!
</ion-content>
```

API

属性	类型	详情
align-title(可选)	`字符串`	标题对齐的位置。可用: 'left', 'right', or 'center'。默认为'center'。

ion-footer-bar

在内容底部添加一个固定的footer栏。

如果应用'bar-subfooter'类，就成了一个subfooter（在上面）。查看footer CSS 文档。

用法

```
<ion-content>
  一些内容！
</ion-content>
<ion-footer-bar align-title="left" class="bar-assertive">
  <div class="buttons">
    <button class="button">左侧按钮</button>
  </div>
  <h1 class="title">Title!</h1>
  <div class="buttons" ng-click="doSomething()">
    <button class="button">右侧按钮</button>
  </div>
</ion-footer-bar>
```

API

属性	类型	详情
align-title(可选)	`字符串`	标题对齐的位置。可选: 'left', 'right', 或 'center'。 默认为 'center'。

Content（内容）

ion-content

授权: `$ionicScrollDelegate`

ionContent指令提供一个易用的内容区域，该区域可以用Ionic的自定义滚动视图进行配置，或浏览器内置的溢出滚动。

在大多数情况下，我们建议使用Ionic的定制滚动功能，有时（出于性能原因）仅用浏览器原生的溢出滚动就足够了，因此我可以轻松地在设置了Ionic滚动和溢出滚动间切换。

你可以用 `ionRefresher` 指令实现拉动刷新，并可以用 `ionInfiniteScroll` 指令实现无限滚动。

用法

```
<ion-content
   [delegate-handle=""]
   [padding=""]
   [scroll=""]
   [overflow-scroll=""]
   [has-bouncing=""]
   [on-scroll=""]
   [on-scroll-complete=""]>
 ...
 </ion-content>
```

API

属性	类型	详情
delegate-handle(可选)	`字符串`	该句柄用于标识带有 `$ionicScrollDelegate` 的滚动视图。
padding(可选)	`布尔值`	是否在内容上添加内填充。在iOS上默认为true，在Android上为false。
scroll(可选)	`布尔值`	是否允许内容滚动。默认为true。
overflow-scroll(可选)	`布尔值`	是否用溢出滚动代替Ionic滚动。
has-bouncing(可选)	`布尔值`	是否允许内容滚动反弹到边缘。iOS上默认为true，Android上默认为false。
on-scroll(可选)	`表达式`	当内容滚动时表现的评估。
on-scroll-complete(可选)	`表达式`	一个滚动动作完成时表现的评估。

ion-refresher

隶属于 `ionContent` 或 `ionScroll`

允许你添加下拉刷新滚动视图。

把它作为 `ionContent` 或 `ionScroll` 元素的第一个子元素。

当刷新完成时，从你的控制器中广播（`$broadcast`）出 'scroll.refreshComplete'事件。

用法

```
<ion-content ng-controller="MyController">
  <ion-refresher
    pulling-text="下拉刷新..."
    on-refresh="doRefresh()">
  </ion-refresher>
  <ion-list>
    <ion-item ng-repeat="item in items"></ion-item>
  </ion-list>
</ion-content>
```

```
angular.module('testApp', ['ionic'])
.controller('MyController', function($scope, $http) {
  $scope.items = [1,2,3];
  $scope.doRefresh = function() {
    $http.get('/new-items')
     .success(function(newItems) {
       $scope.items = newItems;
     })
     .finally(function() {
       // 停止广播ion-refresher
       $scope.$broadcast('scroll.refreshComplete');
     });
  };
});
```

API

属性	类型	详情
on-refresh*(可选)*	`表达式`	当用户下拉到一定程度然后开始刷新时触发。
on-pulling*(可选)*	`表达式`	当用户开始下来刷新时触发。
pulling-icon*(可选)*	`字符串`	当用户下拉时显示的图标。默认: 'ion-arrow-down-c'。
pulling-text*(可选)*	`字符串`	当用户下拉时显示的文字。
refreshing-icon*(可选)*	`字符串`	用户刷新后显示的图标。
refreshing-text*(可选)*	`字符串`	用户刷新后显示的文字。

ion-pane

一个简单的适应内容的容器，无不良影响。在一个元素上添加 'pane'。

用法

```
<ion-pane>
 ...
 </ion-pane>
```

Scroll（滚动）

ion-scroll

授权: `$ionicScrollDelegate`

创建一个包含所有内容的可滚动容器。

用法

```
<ion-scroll
   [delegate-handle=""]
   [direction=""]
   [paging=""]
   [on-refresh=""]
   [on-scroll=""]
   [scrollbar-x=""]
   [scrollbar-y=""]
   [zooming=""]
   [min-zoom=""]
   [max-zoom=""]>
 ...
 </ion-scroll>
```

API

属性	类型	详情
delegate-handle*(可选)*	`字符串`	该句柄利用 `$ionicScrollDelegate` 指定滚动视图。
direction*(可选)*	`字符串`	滚动的方向。 'x' 或 'y'。 默认 'y'。
paging*(可选)*	`布尔值`	分页是否滚动。
on-refresh*(可选)*	`表达式`	调用下拉刷新， 由 `ionRefresher` 触发。
on-scroll*(可选)*	`表达式`	当用户滚动时触发。
scrollbar-x*(可选)*	`布尔值`	是否显示水平滚动条。默认为false。
scrollbar-y*(可选)*	`布尔值`	是否显示垂直滚动条。默认为true。
zooming*(可选)*	`布尔值`	是否支持双指缩放。
min-zoom*(可选)*	`整数`	允许的最小缩放量（默认为0.5）
max-zoom*(可选)*	`整数`	允许的最大缩放量（默认为3）

ion-infinite-scroll

隶属于 `ionContent` 或 `ionScroll`

当用户到达页脚或页脚附近时，ionInfiniteScroll指令允许你调用一个函数 。

当用户滚动的 `距离` 超出底部的内容时，就会触发你指定的 `on-infinite` 。

用法

```
<ion-content ng-controller="MyController">
  <ion-infinite-scroll
    on-infinite="loadMore()"
    distance="1%">
  </ion-infinite-scroll>
</ion-content>
```

```
function MyController($scope, $http) {
  $scope.items = [];
  $scope.loadMore = function() {
    $http.get('/more-items').success(function(items) {
      useItems(items);
      $scope.$broadcast('scroll.infiniteScrollComplete');
    });
  };

  $scope.$on('stateChangeSuccess', function() {
    $scope.loadMore();
  });
}
```

当没有更多数据加载时，就可以用一个简单的方法阻止无限滚动，那就是angular的 `ng-if` 指令:

```
<ion-infinite-scroll
  ng-if="moreDataCanBeLoaded()"
  icon="ion-loading-c"
  on-infinite="loadMoreData()">
</ion-infinite-scroll>
```

API

属性	类型	详情
on-infinite	`表达式`	当滚动到底部时触发的时间。
distance(可选)	`字符串`	从底部滚动到触发on-infinite表达式的距离。默认: 1%。
icon(可选)	`字符串`	当加载时显示的图标。默认: 'ion-loading-d'。

$ionicScrollDelegate

授权控制滚动视图（通过 `ionContent` 和 `ionScroll` 指令创建）。

该方法直接被$ionicScrollDelegate服务触发，来控制所有滚动视图。用 $getByHandle方法控制特定的滚动视图。

用法

```
<body ng-controller="MainCtrl">
  <ion-content>
    <button ng-click="scrollTop()">滚动到顶部!</button>
  </ion-content>
</body>
```

```
function MainCtrl($scope, $ionicScrollDelegate) {
  $scope.scrollTop = function() {
    $ionicScrollDelegate.scrollTop();
  };
}
```

高级用法的例子，用带有两个滚动区域的 `delegate-handle` 来特殊控制。

```
<body ng-controller="MainCtrl">
  <ion-content delegate-handle="mainScroll">
    <button ng-click="scrollMainToTop()">
      滚动内容到顶部!
    </button>
    <ion-scroll delegate-handle="small" style="height: 100px;">
      <button ng-click="scrollSmallToTop()">
        滚动小区域到顶部!
      </button>
    </ion-scroll>
  </ion-content>
</body>
```

```
function MainCtrl($scope, $ionicScrollDelegate) {
  $scope.scrollMainToTop = function() {
    $ionicScrollDelegate.$getByHandle('mainScroll').scrollTop();
  };
  $scope.scrollSmallToTop = function() {
    $ionicScrollDelegate.$getByHandle('small').scrollTop();
  };
}
```

方法

`resize()`

告诉滚动视图重新计算它的容器大小。

`scrollTop([shouldAnimate])`

参数	类型	详情
shouldAnimate*(可选)*	`布尔值`	是否应用滚动动画。

`scrollBottom([shouldAnimate])`

参数	类型	详情
shouldAnimate*(可选)*	`布尔值`	是否应用滚动动画。

`scrollTo(left, top, [shouldAnimate])`

参数	类型	详情
left	`数值`	水平滚动的值。
top	`数值`	垂直滚动的值。
shouldAnimate*(可选)*	`布尔值`	是否应用滚动动画。

`scrollBy(left, top, [shouldAnimate])`

参数	类型	详情
left	`数值`	水平滚动的偏移量。
top	`数值`	垂直滚动的偏移量。
shouldAnimate*(可选)*	`布尔值`	是否应用滚动动画。

`getScrollPosition()`

- 返回: `对象` 滚动到该视图的位置，具有一下属性：
  - `{数值}` `left` 从左侧到用户已滚动的距离(开始为 0)。
  - `{数值}` `top` 从顶部到用户已滚动的距离 (开始为 0)。

`anchorScroll([shouldAnimate])`

告诉滚动视图用一个带有id的滚动元素匹配window.location.hash。

如果没有匹配到元素，它会滚动到顶部。

参数	类型	详情
shouldAnimate*(可选)*	布尔值	是否应用滚动动画。

`getScrollView()`

- 返回: `对象` 匹配具有授权的滚动视图。

`rememberScrollPosition(id)`

当滚动视图被销毁时（用户离开页面），页面最后的滚动位置会被指定的索引保存。

注意：根据一个ion-nav-view将页面和一个视图关联，rememberScrollPosition自动保存它们的滚动。

相关方法：scrollToRememberedPosition, forgetScrollPosition (低)。

在下面的例子中，ion-scroll元素的滚动位置会被保留， 甚至当用户切换开关时。

```
<ion-toggle ng-model="shouldShowScrollView"></ion-toggle>
<ion-scroll delegate-handle="myScroll" ng-if="shouldShowScrollView">
  <div ng-controller="ScrollCtrl">
    <ion-list>
      <ion-item ng-repeat="i in items">{{i}}</ion-item>
    </ion-list>
  </div>
</ion-scroll>
```

```
function ScrollCtrl($scope, $ionicScrollDelegate) {
  var delegate = $ionicScrollDelegate.$getByHandle('myScroll');

  // 这里可以放任何唯一的ID。重点是：要在每次重新创建控制器时
  // 我们要加载当前记住的滚动值。
  delegate.rememberScrollPosition('my-scroll-id');
  delegate.scrollToRememberedPosition();
  $scope.items = [];
  for (var i=0; i<100; i++) {
    $scope.items.push(i);
  }
}
```

参数	类型	详情
id	字符串	保留已滚动位置的滚动视图的id。

`forgetScrollPosition()`

停止保存这个滚动视图的滚动位置。

`scrollToRememberedPosition([shouldAnimate])`

如果这个滚动视图有个和它的滚动位置关联的id（通过调用rememberScrollPosition方法），然后记住那个位置，加载那个位置然后滚动到那个位置。

参数	类型	详情
shouldAnimate*(可选)*	`布尔值`	是否应用滚动动画。

`$getByHandle(handle)`

参数	类型	详情
handle	`字符串`	

- 返回: `delegateInstance` 一个代表性实例就是只控制带有 `delegate-handle` 的滚动视图来匹配给定的句柄。

例如: `$ionicScrollDelegate.$getByHandle('my-handle').scrollTop();`

Tabs（选项卡）

ion-tabs

授权: `$ionicTabsDelegate`

See the Pen by Ionic (@ionic) on CodePen.

带有标签栏的多标签界面的功能是，通过标签切换一组“页面”。

在某个元素上指定任何标签类或 动画类来定义它的外观和感觉。

请查看 `ionTab` 指令文档来了解各个选项卡的更多详情。

注意：不要将ion-tabs置入一个ion-content元素内；它会造成一定的已知CSS错误。

用法

```
<ion-tabs class="tabs-positive tabs-icon-only">

  <ion-tab title="首页" icon-on="ion-ios7-filing" icon-off="ion-ios7-filing-outline">
    <!-- 标签 1 内容 -->
  </ion-tab>

  <ion-tab title="关于" icon-on="ion-ios7-clock" icon-off="ion-ios7-clock-outline">
    <!-- 标签 2 内容 -->
  </ion-tab>

  <ion-tab title="设置" icon-on="ion-ios7-gear" icon-off="ion-ios7-gear-outline">
    <!-- 标签 3 内容 -->
  </ion-tab>

</ion-tabs>
```

API

属性	类型	详情
delegate-handle*(可选)*	`字符串`	该句柄用 `$ionicTabsDelegate` 来标识这些选项卡。

ion-tab

隶属于 `[ionTabs](/js_doc-index-name-ionTabs/)` </small>

包含一个选项卡内容。该内容仅存在于被选中的给定选项卡中。

每个ionTab都有自己的浏览历史。

用法

```
<ion-tab
  title="Tab!"
  icon="my-icon"
  href="#/tab/tab-link"
  on-select="onTabSelected()"
  on-deselect="onTabDeselected()">
</ion-tab>
```

要查看标签栏完整运行的例子，参见 `ionTabs` 文档。

API

属性	类型	详情
title	字符串	选项卡的标题。
href(可选)	字符串	但触碰的时候，该选项卡将会跳转的的链接。
icon(可选)	字符串	选项卡的图标。如果给定值，它将成为ion-on和ion-off的默认值。
icon-on(可选)	字符串	被选中标签的图标。
icon-off(可选)	字符串	没被选中标签的图标。
badge(可选)	表达式	选项卡上的徽章（通常是一个数字）。
badge-style(可选)	表达式	选项卡上徽章的样式（例，tabs-positive ）。
on-select(可选)	表达式	选项卡被选中时触发。
on-deselect(可选)	表达式	选项卡取消选中时触发。
ng-click(可选)	表达式	通常，点击时选项卡会被选中。如果设置了 ng-Click，它将不会被选中。 你可以用$ionicTabsDelegate.select()来指定切换标签。

$ionicTabsDelegate

授权控制 `ionTabs` 指令。

该方法直接调用$ionicTabsDelegate服务，控制所有ionTabs指令。用$getByHandle方法控制具体的ionTabs实例。

用法

```
<body ng-controller="MyCtrl">
  <ion-tabs>

    <ion-tab title="Tab 1">
      你好，标签1！
      <button ng-click="selectTabWithIndex(1)">选择标签2</button>
    </ion-tab>
    <ion-tab title="Tab 2">你好标签2！</ion-tab>

  </ion-tabs>
</body>
```

```
function MyCtrl($scope, $ionicTabsDelegate) {
  $scope.selectTabWithIndex = function(index) {
    $ionicTabsDelegate.select(index);
  }
}
```

方法

`select(index, [shouldChangeHistory])`

选择标签来匹配给定的索引。

参数	类型	详情
index	`数值`	选择标签的索引。
shouldChangeHistory(可选)	`布尔值`	此选项是否应该加载这个标签的浏览历史（如果存在），并使用，或仅加载默认页面。默认为false。提示：如果一个 `ionNavView` 在选项卡里，你可能需要设置它为true。

`selectedIndex()`

- 返回: `数值` 被选中标签的索引，如 -1。

`$getByHandle(handle)`

参数	类型	详情
handle	`字符串`	

- 返回: `delegateInstance` 一个授权实例用 `delegate-handle` 只控制 `ionTabs` 来匹配指定句柄。

例如: `$ionicTabsDelegate.$getByHandle('my-handle').select(0);`

Side Menus（侧栏菜单）

ion-side-menus

授权: `$ionicSideMenuDelegate`

一个容器元素的侧边菜单和主要内容。通过把主要内容区域从一边拖动到另一边，来让左侧或右侧的侧栏菜单进行切换。



Content

要了解更多侧栏菜单的信息，查看文档中的 `ionSideMenuContent` 和 `ionSideMenu` 。

用法

要使用侧栏菜单，添加一个父元素 `<ion-side-menus>` ，一个中间内容 `<ion-side-menu-content>` ，和一个或更多 `<ion-side-menu>` 指令。

```
<ion-side-menus>
  <!-- 中间内容 -->
  <ion-side-menu-content ng-controller="ContentController">
  </ion-side-menu-content>

  <!-- 左侧菜单 -->
  <ion-side-menu side="left">
  </ion-side-menu>

  <!-- 右侧菜单 -->
  <ion-side-menu side="right">
  </ion-side-menu>
</ion-side-menus>
```

```
function ContentController($scope, $ionicSideMenuDelegate) {
  $scope.toggleLeft = function() {
    $ionicSideMenuDelegate.toggleLeft();
  };
}
```

API

属性	类型	详情
delegate-handle*(可选)*	`字符串`	该句柄用来标识带有 `$ionicSideMenuDelegate` 的侧栏菜单。

ion-side-menu-content

隶属于 `ionSideMenus`

一个可见主体内容的容器，同级的一个或多个 `ionSideMenu` 指令。

用法

```
<ion-side-menu-content
  drag-content="true">
</ion-side-menu-content>
```

查看文档中的 `ionSideMenus` ，来了解一个完整侧栏菜单的例子。

API

属性	类型	详情
drag-content(可选)	`布尔值`	内容是否可被拖动。默认为true。

ion-side-menu

隶属于 `ionSideMenus`

一个侧栏菜单的容器，同级的一个 `ionSideMenuContent` 指令。

用法

```
<ion-side-menu
  side="left"
  width="myWidthValue + 20"
  is-enabled="shouldLeftSideMenuBeEnabled()">
</ion-side-menu>
```

完整侧栏菜单的例子，参加文档中的 `ionSideMenus` 。

API

属性	类型	详情
side	`字符串`	侧栏菜单当前在哪一边。可选的值有: 'left' 或 'right'。
is-enabled(可选)	`布尔值`	该侧栏菜单是否可用。
width(可选)	`数值`	侧栏菜单应该有多少像素的宽度。默认为275。

menu-toggle

在一个指定的侧栏中切换菜单。

用法

下面是一个在导航栏内链接的例子。点击此链接会自动打开指定的侧栏菜单。

```
<ion-view>
  <ion-nav-buttons side="left">
   <button menu-toggle="left" class="button button-icon icon ion-navicon"></button>
  </ion-nav-buttons>
 ...
</ion-view>
```

menu-close

关闭当前打开的侧栏菜单。

用法

下面是一个侧栏菜单内链接的例子。点击这个链接会自动关闭当前打开的菜单。

```
<a menu-close href="#/home" class="item">首页</a>
```

$ionicSideMenuDelegate

授权控制 `ionSideMenus` 指令。

该方法直接触发 `$ionicSideMenuDelegate` 服务，来控制所有侧栏菜单。用$getByHandle方法控制特定情况下的ionSideMenus。

用法

```
<body ng-controller="MainCtrl">
  <ion-side-menus>
    <ion-side-menu-content>
      内容!
      <button ng-click="toggleLeftSideMenu()">
        切换左侧侧栏菜单
      </button>
    </ion-side-menu-content>
    <ion-side-menu side="left">
      左侧菜单!
    <ion-side-menu>
  </ion-side-menus>
</body>
```

```
function MainCtrl($scope, $ionicSideMenuDelegate) {
  $scope.toggleLeftSideMenu = function() {
    $ionicSideMenuDelegate.toggleLeft();
  };
}
```

方法

`toggleLeft([isOpen])`

切换左侧侧栏菜单（如果存在）。

参数	类型	详情
isOpen(可选)	`布尔值`	是否打开或关闭菜单。默认：切换菜单。

`toggleRight([isOpen])`

切换右侧侧栏菜单（如果存在）。

参数	类型	详情
isOpen(可选)	`布尔值`	是否打开或关闭菜单。默认：切换菜单。

`getOpenRatio()`

获取打开菜单内容超出菜单宽度的比例。比如，一个宽度为100px的菜单被宽度为50px以50%的比例打开，将会返回一个比例值为0.5。

- 返回: `浮点` 0 表示没被打开，如果左侧菜单处于已打开或正在打开为0 到 1，如果右侧菜单处于已打开或正在打开为0 到-1。

`isOpen()`

- 返回: `布尔值` 无论左侧或右侧菜单是否已经打开。

`isOpenLeft()`

- 返回: `布尔值` 左侧菜单是否已经打开。

`isOpenRight()`

- 返回: `布尔值` 右侧菜单是否已经打开。

`canDragContent([canDrag])`

参数	类型	详情
canDrag(可选)	`布尔值`	设置是否可以拖动内容打开侧栏菜单。

- 返回: `布尔值` 是否可以拖动内容打开侧栏菜单。

`$getByHandle(handle)`

参数	类型	详情
handle	`字符串`	

- 返回: `delegateInstance` 一个代表性的实例是用 `delegate-handle` 只控制 `ionSideMenus` 指令来匹配指定的句柄。

例如: `$ionicSideMenuDelegate.$getByHandle('my-handle').toggleLeft();`

Navigation（导航）

ion-nav-view

See the Pen by Ionic (@ionic) on CodePen.

当用户在你的app中浏览时，Ionic能够保持检测他们的浏览历史。通过了解他们的浏览历史，向左或向右滑动时可以正确的在视图间转换，或不转换。一个额外的好处是Ionic的导航系统具有可以管理多个历史记录的能力。

Ionic利用AngularUI路由模块，使应用程序接口可以组织成不同的“状态”。例如Angular的核心$route服务，利用URL控制视图。然而，AngularUI路由提供了一个更强大的状态管理，即状态可以被命名，嵌套， 以及合并视图，允许一个以上模板呈现在同一个页面。此外，每个状态无需绑定到一个URL，并且数据可以更灵活地推送到每个状态。

用ionNavView指令在你的app中渲染模版。每个模板都是状态的一部分。状态通常映射到一个url上，然后用angular-ui-router定义程序（查看它们的文档，记下用ion-nav-view替换ui-view的例子）。

用法

在这个例子中，我们将创建一个应用程序中包含不同状态的导航视图。

要做到这一点，在我们的标签中，用ionNavView顶层指令。要显示一个页眉，我们利用，当我们通过导航堆栈导航时，就会用 `ionNavBar` 指令更新。

你可以在navView的 `动画` 属性上应用任何动画类来给它添加页面动画。

建议的页面过渡： 'slide-left-right', 'slide-left-right-ios7', 'slide-in-up'。

```
<ion-nav-bar></ion-nav-bar>
<ion-nav-view animation="slide-left-right">
  <!-- 中间内容 -->
</ion-nav-view>
```

接下来，我们需要设置被渲染的状态。

```
var app = angular.module('myApp', ['ionic']);
app.config(function($stateProvider) {
  $stateProvider
  .state('index', {
    url: '/',
    templateUrl: 'home.html'
  })
  .state('music', {
    url: '/music',
    templateUrl: 'music.html'
  });
});
```

然后在app启动时，$stateProvider就会检查url，检查它的索引匹配状态，然后尝试将home.html加载到 `<ion-nav-view>` 内。

页面由指定的URL加载。在Angular中有一个简单的方式就是把它们直接放到你的HTML文件，并用 `<script type="text/ng-template">` 语法。因此，这是一种把home.html载入到app中的一种方式:

```
<script id="home" type="text/ng-template">
  <!-- ion-view的标题会在导航栏显示 -->
  <ion-view title="'Home'">
    <ion-content ng-controller="HomeCtrl">
      <!-- 页面的内容 -->
      <a href="#/music">跳转到音乐页面！!</a>
    </ion-content>
  </ion-view>
</script>
```

这么做那也是极好的，因为缓存了模板，加载速度非常快，不必从网络上再获取。

请访问AngularUI 路由文档了解详情。下面是一个很棒的AngularUI路由视频，可以帮助了解它的所有运作情况：

API

属性	类型	详情
name(可选)	`字符串`	一个视图的名字。这个名字应该是在相同的状态下其他视图中唯一的。你可以在不同的状态中有相同名称的视图。欲了解详细信息，查看ui-router的 ui-view 文档。

ion-view

隶属于 `ionNavView`

一个内容的容器，用来告诉一个当前视图的父 `ionNavBar` 。

用法

下面是一个带有“我的页面”标题的导航栏载入页面的例子。

```
<ion-nav-bar></ion-nav-bar>
<ion-nav-view class="slide-left-right">
  <ion-view title="我的页面">
    <ion-content>
      你好！
    </ion-content>
  </ion-view>
</ion-nav-view>
```

API

属性	类型	详情
title(可选)	`字符串`	显示在父 `ionNavBar` 的标题。
hide-back-button(可选)	`布尔值`	默认情况下，是否在父 `ionNavBar` 隐藏后退按钮。
hide-nav-bar(可选)	`布尔值`	默认情况下，是否隐藏父 `ionNavBar` 。

ion-nav-bar

授权: `$ionicNavBarDelegate`

如果我们有一个 `ionNavView` 指令，我们也可以创建一个 `<ion-nav-bar>` ，它会创建一个顶部工具栏，当程序状态改变时更新。

我们在里面放入一个 `ionNavBackButton` 来添加一个后退按钮。

用 `ionNavButtons` 根据当前可用的视图添加按钮。

在一个元素上指定一个动画类，来启用更换标题的动画（建议: 'nav-title-slide-ios7'）

用法

```
<body ng-app="starter">
  <!-- 当我们浏览时，导航栏会随之更新。 -->
  <ion-nav-bar class="bar-positive nav-title-slide-ios7">
  </ion-nav-bar>

  <!-- 初始化时渲染视图模板 -->
  <ion-nav-view></ion-nav-view>
</body>
```

API

属性	类型	详情
delegate-handle(可选)	`字符串`	该句柄用 `$ionicNavBarDelegate` 标识此导航栏。
align-title(可选)	`字符串`	导航栏标题对齐的位置。可用： 'left', 'right', 'center'。默认为 'center'。

其他用法

除此之外，你可以将ion-nav-bar放到每个单独视图的ion-view元素里面。它允许你把整个导航栏，而不仅是它的内容，改变每个视图的过渡。

这类似于把header栏嵌入到你的ion-view中，此外它有导航栏的所有功能。

如果你这样做，只需把导航按钮放在导航栏里面；而不需要用 `<ion-nav-buttons>` 。

```
<ion-nav-bar class="bar-positive">
  <ion-nav-back-button>
    返回
  </ion-nav-back-button>
  <div class="buttons right-buttons">
    <button class="button">
      右侧按钮
    </button>
  </div>
</ion-nav-bar>
<ion-view title="我的标题">
</ion-view>
```

改进该文档

ion-nav-buttons

隶属于 `ionNavView`

在 `ionView` 内的 `ionNavBar` 上用ionNavButtons设置按钮。

你设置的任何按钮都将被放置在导航栏的相应位置，当用户离开父视图时会被销毁。

用法

```
<ion-nav-bar>
</ion-nav-bar>
<ion-nav-view>
  <ion-view>
    <ion-nav-buttons side="left">
      <button class="button" ng-click="doSomething()">
        我是一个在导航栏左侧的按钮！
      </button>
    </ion-nav-buttons>
    <ion-content>
      这里是一些内容！
    </ion-content>
  </ion-view>
</ion-nav-view>
```

API

属性	类型	详情
side	`字符串`	在父 `ionNavBar` 中按钮放置的位置。 可用: 'left' 或 'right'。

ion-nav-back-button

隶属于 `ionNavBar`

在一个 `ionNavBar` 中创建一个按钮。

当用户在当前导航堆栈能够后退时，将显示后退按钮。

默认情况下，当点击后退按钮时。如果你想了解更高级的行为，请参阅下面的例子。

用法

默认按钮动作:

```
<ion-nav-bar>
  <ion-nav-back-button class="button-clear">
    <i class="ion-arrow-left-c"></i> 后退
  </ion-nav-back-button>
</ion-nav-bar>
```

自定义点击动作，用 `$ionicNavBarDelegate`:

```
<ion-nav-bar ng-controller="MyCtrl">
  <ion-nav-back-button class="button-clear"
    ng-click="canGoBack && goBack()">
    <i class="ion-arrow-left-c"></i> 后退
  </ion-nav-back-button>
</ion-nav-bar>
```

```
function MyCtrl($scope, $ionicNavBarDelegate) {
  $scope.goBack = function() {
    $ionicNavBarDelegate.back();
  };
}
```

在后退按钮上显示上一个标题，也用 `$ionicNavBarDelegate`。

```
<ion-nav-bar ng-controller="MyCtrl">
  <ion-nav-back-button class="button-icon">
    <i class="icon ion-arrow-left-c"></i>{{getPreviousTitle() || 'Back'}}
  </ion-nav-back-button>
</ion-nav-bar>
```

```
function MyCtrl($scope, $ionicNavBarDelegate) {
  $scope.getPreviousTitle = function() {
    return $ionicNavBarDelegate.getPreviousTitle();
  };
}
```

nav-clear

nav-clear一个当点击视图上的元素时用到的属性指令，比如一个 `<a href>` 或者一个 `<button ui-sref>` 。

当点击时，nav-clear将会导致给定的元素，禁止下一个视图的转换。这个指令很有用，比如，侧栏菜单内的链接。

用法

下面是一个侧栏菜单内添加了nav-clear指令的一个链接。点击该链接将禁用视图间正常进行的任何动画。

```
<a nav-clear menu-close href="#/home" class="item">首页</a>
```

$ionicNavBarDelegate

授权控制 `ionNavBar` 指令。

用法

```
<body ng-controller="MyCtrl">
  <ion-nav-bar>
    <button ng-click="setNavTitle('香蕉')">
      设置标题为香蕉！
    </button>
  </ion-nav-bar>
</body>
```

```
function MyCtrl($scope, $ionicNavBarDelegate) {
  $scope.setNavTitle = function(title) {
    $ionicNavBarDelegate.setTitle(title);
  }
}
```

方法

`back([event])`

在浏览历史中后退。

参数	类型	详情
event(可选)	`DOMEvent`	事件对象（如来自点击事件）

`align([direction])`

带有按钮的标题对齐到指定的方向。

参数	类型	详情
direction(可选)	`字符串`	标题文字对齐的方向。可用: 'left', 'right', 'center'。 默认: 'center'。

`showBackButton([show])`

设置或获取 `ionNavBackButton` 是否显示（如果它存在的话）。

参数	类型	详情
show(可选)	`布尔值`	后退按钮是否显示。

- 返回: `布尔值` 后退按钮是否显示。

`showBar(show)`

设置或获取 `ionNavBar` 是否显示。

参数	类型	详情
show	`布尔值`	导航栏是否显示。

- 返回: `布尔值` 导航栏是否显示。

`setTitle(title)`

为 `ionNavBar` 设置标题。

参数	类型	详情
title	`字符串`	显示新标题。

`changeTitle(title, direction)`

改变标题，指定的一个过渡的方向，显示新标题，隐藏旧标题。

参数	类型	详情
title	`字符串`	显示新标题。
direction	`字符串`	过渡显示新标题的方向。可用： 'forward', 'back'。

`getTitle()`

- 返回: `字符串` 获取当前导航栏的标题

`getPreviousTitle()`

- 返回: `字符串` 导航栏的上一个标题。

`$getByHandle(handle)`

参数	类型	详情
handle	`字符串`	

- 返回: `delegateInstance` 用给定的delegate-handle句柄只控制导航栏的一个授权实例。

例如: `$ionicNavBarDelegate.$getByHandle('myHandle').setTitle('newTitle')`

Lists（列表）

ion-list

授权: `$ionicListDelegate`

列表是一个应用广泛在几乎所有移动app中的界面元素，可以包含的内容范围从基本文字到按钮，开关，图标和缩略图在内所有内容。

包含列表项的列表以及列表项自身都可以是任何的HTML元素。容器元素需要 `list` 类，并且每个列表项需要 `item` 类。

然而，使用ionList和ionItem可以很容易的支持各种交互方式，比如，滑动编辑，拖动排序，以及删除项。

相关阅读: `ionItem` , `ionOptionButton` `ionReorderButton` , `ionDeleteButton` , `list CSS documentation` .

用法

基本用法:

```
<ion-list>
  <ion-item ng-repeat="item in items">
    Hello, {{item}}!
  </ion-item>
</ion-list>
```

高级用法: 缩略图，删除按钮，重新排序，滑动

```
<ion-list ng-controller="MyCtrl"
          show-delete="shouldShowDelete"
          show-reorder="shouldShowReorder"
          can-swipe="listCanSwipe">
  <ion-item ng-repeat="item in items"
            class="item-thumbnail-left">

    <img ng-src="{{item.img}}">
    <h2>{{item.title}}</h2>
    <p>{{item.description}}</p>
    <ion-option-button class="button-positive"
                       ng-click="share(item)">
      分享
    </ion-option-button>
    <ion-option-button class="button-info"
                       ng-click="edit(item)">
      编辑
    </ion-option-button>
    <ion-delete-button class="ion-minus-circled"
                       ng-click="items.splice($index, 1)">
    </ion-delete-button>
    <ion-reorder-button class="ion-navicon"
                        on-reorder="reorderItem(item, $fromIndex, $toIndex)">
    </ion-reorder-button>

  </ion-item>
</ion-list>
```

API

属性	类型	详情
delegate-handle(可选)	字符串	该句柄定义带有 `$ionicListDelegate` 的列表。
show-delete(可选)	布尔值	列表项的删除按钮当前是显示还是隐藏。
show-reorder(可选)	布尔值	列表项的排序按钮当前是显示还是隐藏。
can-swipe(可选)	布尔值	列表项是否被允许滑动显示选项按钮。默认：true。

ion-item

隶属于 `ionList`

用法

```
<ion-list>
  <ion-item>你好!</ion-item>
</ion-list>
```

ion-delete-button

隶属于 `ionItem`

用法

```
<ion-list show-delete="shouldShowDelete">
  <ion-item>
    <ion-delete-button class="ion-minus-circled"></ion-delete-button>
    Hello, 列表项!
  </ion-item>
</ion-list>
<ion-toggle ng-model="shouldShowDelete">
  显示删除?
</ion-toggle>
```

ion-reorder-button

隶属于 `ionItem`

用法

```
<ion-list ng-controller="MyCtrl">
  <ion-item ng-repeat="item in items">
    项
    <ion-reorder-button class="ion-navicon"
                        on-reorder="moveItem(item, $fromIndex, $toIndex)">
    </ion-reorder>
  </ion-item>
</ion-list>
```

```
function MyCtrl($scope) {
  $scope.items = [1, 2, 3, 4];
  $scope.moveItem = function(item, fromIndex, toIndex) {
    //把该项移动到数组中
    $scope.items.splice(fromIndex, 1);
    $scope.items.splice(toIndex, 0, item);
  };
}
```

API

属性	类型	详情
on-reorder(可选)	`表达式`	当一项被重新排序时调用表达式。给定参数：$fromIndex, $toIndex。

ion-option-button

隶属于 `ionItem`

用法

```
<ion-list>
  <ion-item>
    我喜欢小猫!
    <ion-option-button class="button-positive">分享</ion-option-button>
    <ion-option-button class="button-assertive">编辑</ion-option-button>
  </ion-item>
</ion-list>
```

collection-repeat

`collection-repeat` 是一个允许你渲染数千项列表，并且性能几乎不会减弱的指令。

示例：

该指令只渲染屏幕可见区域的列表。因此如果你有1000条列表项目，只有10条呈现在你的屏幕上， collection-repeat 只会渲染当前滚动位置的十条DOM。

当使用collection-repeat时，请记住一下几点：

1. collection-repeat处理的数据必须是一个数组。
2. 你必须明确的告诉该指令，在DOM中你的项使用是多大的指令属性。允许像素值或百分百（见下文）。
3. 被渲染的元素将被绝对定位：确保你的CSS正常运行（见下文）。
4. 保持重复的HTML元素尽可能的简单。当用户向下滚动时，元素会被延迟渲染。因此，你的元素越复杂，在用户滚动的过程中按需编译就越会导致“卡屏”。
5. 在屏幕的每一行，你渲染的元素越多，滚动就越可能变慢。建议列表元素的栅格保持在3

列以内。例如，如果你有一个图库只需把它们的宽度设置为33%。

6. 每个 collection-repeat列表都会占据它的所有父滚动视图的空间。如果你想在一个页面上有多个列表，就把每个列表放在它自己的 ionScroll容器内。
7. 你不应该在带有collection-repeat列表的ion-content或ion-scroll元素上使用ng-show 和ng-hide指令。 ng-show 和 ng-hide 在内容样式上应用 css规则 `display: none` ，导致可见滚动视图内容的宽度和高度为0。因此，collection-repeat会渲染刚刚被取消隐藏的元素。

用法

基本用法（单行项）

注意两点：我们用 ng-style来设置项的高度以匹配我们的重复项的高度。此外， 我们添加了一个css规则使我们的item拉伸以适应全屏（因为它是绝对定位的）。

```
<ion-content ng-controller="ContentCtrl">
  <div class="list">
    <div class="item my-item"
      collection-repeat="item in items"
      collection-item-width="'100%'"
      collection-item-height="getItemHeight(item, $index)"
      ng-style="{height: getItemHeight(item, $index)}">
      {{item}}
    </div>
  </div>
</div>
```

```
function ContentCtrl($scope) {
  $scope.items = [];
  for (var i = 0; i < 1000; i++) {
    $scope.items.push('Item ' + i);
  }

  $scope.getItemHeight = function(item, index) {
    //使索引项平均都有10px高，例如
    return (index % 2) === 0 ? 50 : 60;
  };
}
```

```
.my-item {
  left: 0;
  right: 0;
}
```

栅格用法（每行三项）

```
<ion-content>
  <div class="item item-avatar my-image-item"
    collection-repeat="image in images"
    collection-item-width="'33%'"
    collection-item-height="'33%'">
    <img ng-src="">
  </div>
</ion-content>
```

```
.my-image-item {
  height: 33%;
  width: 33%;
}
```

API

`collection-repeat`

`表达式`

该表达式表明如何枚举一个集合。当前支持一下格式：

- `变量表达式` – 其中变量是用户定义的循环变量，`表达式` 是一个给出了集合进行枚举的范围表达式。

  比如: `album in artist.albums` 。

- `通过tracking_expression跟踪表达式变量` – 你也可以提供一个可选的跟踪功能可以将集合中的对象与DOM元素关联起来。如果没有指定跟踪功能，collection-repeat通过在集合中标识来关联元素。用一个以上的跟踪功能来解决同一个问题是错误的。（这意味着两个不同的对象被映射到同一个DOM元素上，那样是不行的。）过滤器应被应用到表达式，特定跟踪表达式之前。

  比如: `item in items` 等同于'item in items track by $id(item)'。 这意味着DOM元素会被数组中标识的项关联。

  比如: `item in items track by $id(item)` 。在数组的每一项中一个内置的 `$id()` 函数可以被用来分配一个唯一的 `$$hashKey` 属性。然后这个属性作为一个关联DOM元素的关键，通过标识数组中的相应的项。在数组中移动同一个项与在DOM中移动DOM元素的方式一样。

  比如: 当数据来自数据库时，`item in items track by item.id` 是一个典型的模式。在这种情况下，对象标识并不重要。重要两个对象的 `id` 属性相同，那么它们被认为是等效的。

  比如: `item in items | filter:searchText track by item.id` 是一种模式，可用来在项上应用一个过滤器，与一个跟踪表达式相结合。

`collection-item-width`

`表达式`

重复元素的宽度。可以是一个数字（以像素为单位）或一个百分百。

`collection-item-height`

`表达式`

重复元素的高度。可以是一个数字（以像素为单位）或一个百分百。

$ionicListDelegate

授权控制 `ionList` 指令。

当$ionicListDelegate服务控制所有列表时，会直接调用该方法。用 $getByHandle方法控制特定的ionList实例。

用法

```
<ion-content ng-controller="MyCtrl">
  <button class="button" ng-click="showDeleteButtons()"></button>
  <ion-list>
    <ion-item ng-repeat="i in items">>
      Hello, {{i}}!
      <ion-delete-button class="ion-minus-circled"></ion-delete-button>
    </ion-item>
  </ion-list>
</ion-content>
```

```
function MyCtrl($scope, $ionicListDelegate) {
  $scope.showDeleteButtons = function() {
    $ionicListDelegate.showDelete(true);
  };
}
```

方法

`showReorder([showReorder])`

参数	类型	详情
showReorder*(可选)*	`布尔值`	设置是否显示该列表的排序按钮。

- 返回: `布尔值` 排序按钮是否显示。

`showDelete([showReorder])`

参数	类型	详情
showReorder*(可选)*	`布尔值`	设置是否显示该列表的删除按钮。

- 返回: `布尔值` 删除按钮是否显示。

`canSwipeItems([showReorder])`

参数	类型	详情
showReorder*(可选)*	`布尔值`	设置该列表是否可以切换显示选项按钮。

- 返回: `布尔值` 该列表是否可以切换显示选项按钮。

`closeOptionButtons()`

关闭所有被打开的列表选项按钮。

`$getByHandle(handle)`

参数	选项	详情
handle	`字符串`	

- 返回: `delegateInstance` 一个只控制带有 `delegate-handle` 匹配给定句柄的 `ionList` 授权实例。

例如: `$ionicListDelegate.$getByHandle('my-handle').showReorder(true);`

Form Inputs（表单）

ion-checkbox

ion-checkbox和HTML复选框相比没什么不同，除了它的样式不同。

复选框的行为类似于任何AngularJS 复选框。

Usage

```
<ion-checkbox ng-model="isChecked">复选框标签</ion-checkbox>
```

ion-radio

radio指令和HTMLradio相比没什么不同，除了它的样式不一样。

Radio的行为类似于任何AngularJS radio。

用法

```
<ion-radio ng-model="choice" ng-value="A">Choose A</ion-radio>
<ion-radio ng-model="choice" ng-value="B">Choose B</ion-radio>
<ion-radio ng-model="choice" ng-value="C">Choose C</ion-radio>
```

ion-toggle

一个切换效果就是一个绑定一个给定布尔值模型的开关动画。

允许拖动开关的按钮。

切换的行为类似与任何AngularJS 复选框。

用法

下面是被连接到 `airplaneMode` 模块并且关联 `toggle-calm` CSS类内部元件的一个切换指令的例子。

```
<ion-toggle ng-model="airplaneMode" toggle-class="toggle-calm">Airplane Mode</ion-toggle>
```

API

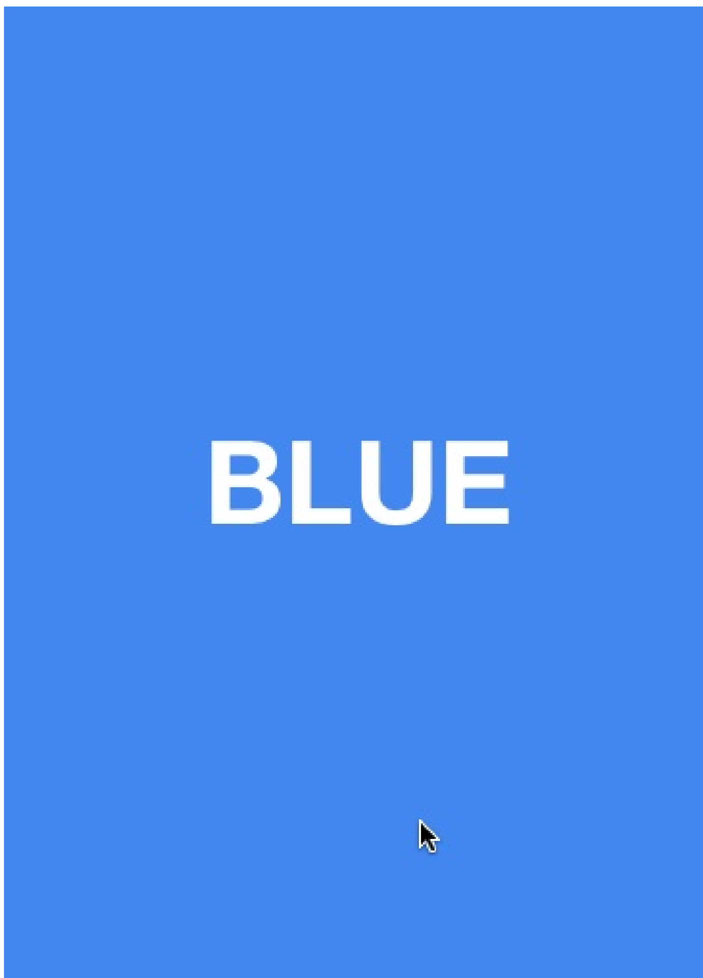
属性	类型	详情
toggle-class*(可选)*	`字符串`	通过该指令在内部设置CSS类创建 `label.toggle` 元素。

Slide Box（滑动框）

ion-slide-box

授权: `$ionicSlideBoxDelegate`

滑动框是一个包含多页容器的组件，每页滑动或拖动切换：



用法

```
<ion-slide-box>
  <ion-slide>
    <div class="box blue"><h1>BLUE</h1></div>
  </ion-slide>
  <ion-slide>
    <div class="box yellow"><h1>YELLOW</h1></div>
  </ion-slide>
  <ion-slide on-slide-changed="slideHasChanged(index)">
    <div class="box pink"><h1>PINK</h1></div>
  </ion-slide>
</ion-slide-box>
```

API

属性	类型	详情
delegate-handle(可选)	字符串	该句柄用 `$ionicSlideBoxDelegate` 来标识这个滑动框。
does-continue(可选)	布尔值	滑动框是否自动滑动。
slide-interval(可选)	数字	等待多少毫秒开始滑动（如果继续则为true）。默认为4000。
show-pager(可选)	布尔值	滑动框的页面是否显示。
pager-click(可选)	表达式	当点击页面时，触发该表达式（如果shou-pager为true）。传递一个'索引'变量。
on-slide-changed(可选)	表达式	当滑动时，触发该表达式。传递一个'索引'变量。
active-slide(可选)	表达式	将模型绑定到当前滑动框。

$ionicSlideBoxDelegate

授权控制 `ionSlideBox` 指令。

当$ionicSlideBoxDelegate服务控制所有滑动框时触发该方法。用 $getByHandle方法控制特定的滑动框实例。

用法

```
<body ng-controller="MyCtrl">
  <ion-slide-box>
    <ion-slide>
      <div class="box blue">
        <button ng-click="nextSlide()">下一个滑块!</button>
      </div>
    </ion-slide>
    <ion-slide>
      <div class="box red">
        滑块 2!
      </div>
    </ion-slide>
  </ion-slide-box>
</body>
```

```
function MyCtrl($scope, $ionicSlideBoxDelegate) {
  $scope.nextSlide = function() {
    $ionicSlideBoxDelegate.next();
  }
}
```

方法

`update()`

更新滑动框（例如，用带有ng-repeat的Angular，调整它里面的元素）。

`slide(to, [speed])`

参数	类型	详情
to	`数字`	滑动的索引。
speed*(可选)*	`数字`	滑动切换的毫秒数。

`enableSlide([shouldEnable])`

参数	类型	详情
shouldEnable*(可选)*	`布尔值`	是否启用滑动框的滑动功能。

- 返回: `布尔值` 是否启用滑动。

`previous()`

跳转到上一个滑块。如果在开始滑块，就循环。

`next()`

跳转到下一个滑块。如果在结尾就循环。

`stop()`

停止滑动。滑动框将不会再被滑动，直到再次启用。

`currentIndex()`

- 返回: 当前滑块的索引数值。

`slidesCount()`

- 返回: 当前滑块的数目。

`$getByHandle(handle)`

参数	类型	详情
handle	`字符串`	

- 返回: `delegateInstance` 一个只控制带有 `delegate-handle` 的 `ionSlideBox` 指令的授权实例来匹配给定句柄。

例如: `$ionicSlideBoxDelegate.$getByHandle('my-handle').stop();`

Modal（模型）

$ionicModal

模型是一个内容面板，可以临时越过用户的主视图。通常用于选择或编辑一个项。注意，你需要把模型的内容放入一个带有 `modal` 类的div内。

用法

```
<script id="my-modal.html" type="text/ng-template">
  <div class="modal">
    <ion-header-bar>
      <h1 class="title">我的模型标题</h1>
    </ion-header-bar>
    <ion-content>
      Hello!
    </ion-content>
  </div>
</script>
```

```
angular.module('testApp', ['ionic'])
.controller('MyController', function($scope, $ionicModal) {
  $ionicModal.fromTemplateUrl('modal.html', {
    scope: $scope,
    animation: 'slide-in-up'
  }).then(function(modal) {
    $scope.modal = modal;
  });
  $scope.openModal = function() {
    $scope.modal.show();
  };
  $scope.closeModal = function() {
    $scope.modal.hide();
  };
  //当我们用到模型时，清除它！
  $scope.$on('$destroy', function() {
    $scope.modal.remove();
  });
  // 当隐藏的模型时执行动作
  $scope.$on('modal.hide', function() {
    // 执行动作
  });
  // 当移动模型时执行动作
  $scope.$on('modal.removed', function() {
    // 执行动作
  });
});
```

方法

`fromTemplate(templateString, options)`

参数	类型	详情
templateString	`字符串`	模板的字符串作为模型的内容。
options	`对象`	传递ionicModal#initialize方法的选项。

- 返回: `对象` 一个 `ionicModal` 控制器的实例。

`fromTemplateUrl(templateUrl, options)`

参数	类型	详情
templateUrl	`字符串`	载入模板的url。
options	`对象`	通过ionicModal#initialize方法传递对象。

- 返回: `promise` 用 `ionicModal` 控制器的一个实例加以解决的承诺。

ionicModal

由 `$ionicModal` 服务实例化。

提示：当你完成每个模块清除时，确保调用remove()方法，以避免内存泄漏。

注意：一个模块从它的初始范围广播出 'modal.shown' 和 'modal.hidden' ，把自身作为一个参数来传递。

方法

`initialize(可选)`

创建一个新的模型控制器示例。

参数	类型	详情
options	`对象`	选项对象

一个选项对象具有一下属性：

- `{object=}` `范围` 子类的范围。默认：创建一个$rootScope子类。
- `{string=}` `动画` 带有显示或隐藏的动画。默认：'slide-in-up'
- `{boolean=}` `第一个输入框获取焦点` 当显示时，模型的第一个输入元素是否自动获取焦点。默认：false。
- `{boolean=}` `backdropClickToClose` 点击背景时是否关闭模型。默认：true。

`show()`

显示这个模型实例。

- 返回: `凭证` 当模型完成动画进入时，解决的一种凭证。

`hide()`

隐藏模型实例

- 返回: `凭证` 当模型完成动画退出时，解决的一种凭证。

`remove()`

从DOM中移动并清除这个模型实例。

- 返回: `凭证` 当模型完成动画退出时，解决的一种凭证。

`isShown()`

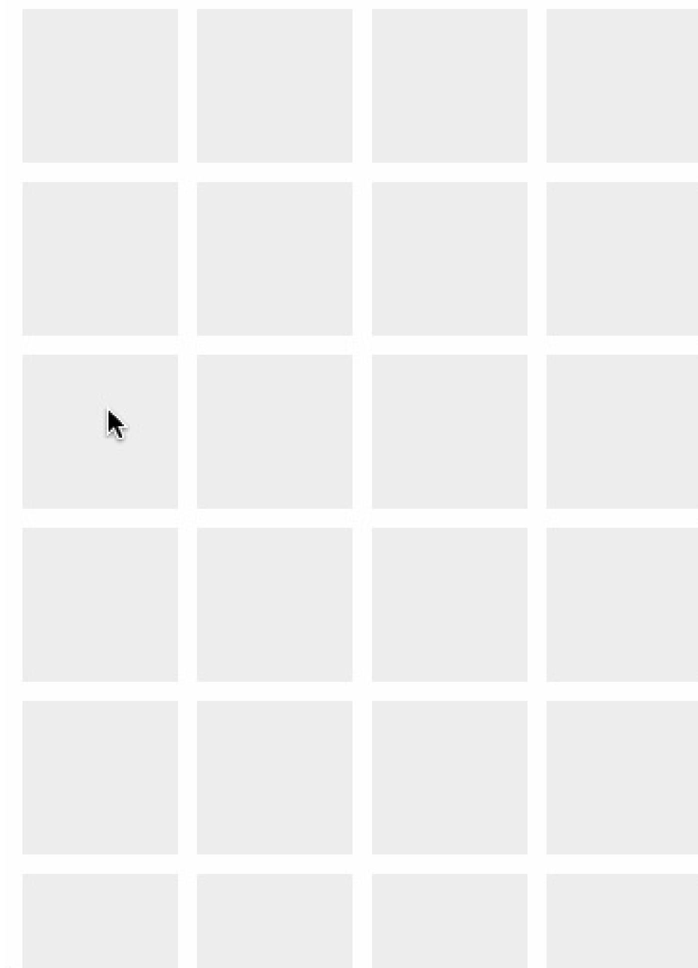
- 返回: 模型当前是否显示。

Action Sheet（操作表）

$ionicActionSheet

该操作表是一个向上滑动的面板，用户可以从一系列选项中选择。危险的操作以红色突出显示。

有简便的方式可以取消操作表，例如点击背景，在桌面电脑测试时，按ESC键也可以。



用法

在你的代码中触发一个操作表，在angular控制器中用 $ionicActionSheet服务：

```
angular.module('mySuperApp', ['ionic'])
.controller(function($scope, $ionicActionSheet) {

 // 点击按钮触发，或一些其他的触发条件
 $scope.show = function() {

   // 显示操作表
   $ionicActionSheet.show({
     buttons: [
       { text: '<b>Share</b> This' },
       { text: 'Move' },
     ],
     destructiveText: 'Delete',
     titleText: 'Modify your album',
     cancelText: 'Cancel',
     buttonClicked: function(index) {
       return true;
     }
   });

 };
});
```

方法

`show(opts)`

加载并返回一个新的操作表。

针对操作表的一种新的隔离范围将被创建，新的元素会附加进body内。

参数	类型	详情
opts	`对象`	操作表的选项。

属性：

- `[Object]` `按钮` 显示的按钮。每个按钮都是一个带有 `文字` 的对象。
- `{string}` `标题文字` 在操作表上显示的标题。
- `{string=}` `取消文字` 操作表上'取消'按钮的文字。
- `{string=}` `警告文字` 操作表上'警告'的文字。
- `{function=}` `取消` 当点击取消按钮或点击背景时触发。
- `{function=}` `点击按钮` 当非警告按钮之一被点击时触发，带有索引的按钮被点击和按钮对象。返回true则关闭操作表，或false则保持打开。
- `{function=}` `点击警告按钮` 当警告按钮被点击时触发。返回true则关闭操作表，或false则保持打开。

Popup（弹出窗口）

$ionicPopup

See the Pen by Ionic (@ionic) on CodePen.

Ionic弹窗服务允许程序创建、显示弹出窗口，需要用户继续响应。

弹窗系统支持更多灵活的构建`alert()`，`prompt()`和`confirm()`功能版本，以及用户习惯，除了允许查看完全自定义内容的的弹窗。

用法

一些基本的例子，查看下文了解所有可用的选项详情。

```
angular.module('mySuperApp', ['ionic'])
.controller(function($scope, $ionicPopup, $timeout) {

 // 触发一个按钮点击，或一些其他目标
 $scope.showPopup = function() {
   $scope.data = {}

   // 一个精心制作的自定义弹窗
   var myPopup = $ionicPopup.show({
     template: '<input type="password" ng-model="data.wifi">',
     title: 'Enter Wi-Fi Password',
     subTitle: 'Please use normal things',
     scope: $scope,
     buttons: [
       { text: 'Cancel' },
       {
         text: '<b>Save</b>',
         type: 'button-positive',
         onTap: function(e) {
           if (!$scope.data.wifi) {
             //不允许用户关闭，除非他键入wifi密码
             e.preventDefault();
           } else {
             return $scope.data.wifi;
           }
         }
       },
     ]
   });
   myPopup.then(function(res) {
     console.log('Tapped!', res);
   });
   $timeout(function() {
      myPopup.close(); //由于某种原因3秒后关闭弹出
   }, 3000);

   // 一个确认对话框
   $scope.showConfirm = function() {
     var confirmPopup = $ionicPopup.confirm({
       title: 'Consume Ice Cream',
       template: 'Are you sure you want to eat this ice cream?'
     });
     confirmPopup.then(function(res) {
       if(res) {
         console.log('You are sure');
       } else {
         console.log('You are not sure');
       }
     });
   };

   // 一个提示对话框
   $scope.showAlert = function() {
     var alertPopup = $ionicPopup.alert({
       title: 'Don\'t eat that!',
       template: 'It might taste good'
     });
     alertPopup.then(function(res) {
       console.log('Thank you for not eating my delicious ice cream cone');
     });
   };
 };
});
```

方法

`show(可选)`

显示一个复杂的对话框。 这是一个所有弹窗的主体显示功能。

一个带有 `按钮` 组的复杂弹窗，每个按钮带有一个 `文本` 和 `类型` 字段，此外还有一个 `onTap` 功能。当点击弹窗上的相关按钮，会触发 `onTap` 函数，默认会关闭弹窗，处理弹窗的返回值。如果你想阻止默认动作，点击按钮保持打开弹窗，当点击一个事件时，触发 `event.preventDefault()` 。详见下文。

参数	类型	详情
options	`object`	新弹窗的选项的表现形式

```
{
  title: '', // String. 弹窗的标题。
  subTitle: '', // String (可选)。弹窗的子标题。
  template: '', // String (可选)。放在弹窗body内的html模板。
  templateUrl: '', // String (可选)。在弹窗body内的html模板的URL。
  scope: null, // Scope (可选)。一个链接到弹窗内容的scope（作用域）。
  buttons: [{ //Array[Object] (可选)。放在弹窗footer内的按钮。
    text: 'Cancel',
    type: 'button-default',
    onTap: function(e) {
      // 当点击时，e.preventDefault() 会阻止弹窗关闭。
      e.preventDefault();
    }
  }, {
    text: 'OK',
    type: 'button-positive',
    onTap: function(e) {
      // 返回的值会导致处理给定的值。
      return scope.data.response;
    }
  }]
}
```

- 返回: `object` 当关闭弹窗时，处理一个promise。有一个附加的 `关闭` 函数，用于利用程序关闭弹窗。

`alert(可选)`

显示一个带有一段信息和一个用户可以点击关闭弹窗的按钮的简单提示弹窗。

参数	类型	**Details**
options	`object`	显示提示的选项形式

```
{
  title: '', // String. 弹窗的标题。
  subTitle: '', // String (可选)。弹窗的子标题。
  template: '', // String (可选)。放在弹窗body内的html模板。
  templateUrl: '', // String (可选)。 放在弹窗body内的html模板的URL。
  okText: '', // String (默认: 'OK')。OK按钮的文字。
  okType: '', // String (默认: 'button-positive')。OK按钮的类型。
}
```

- 返回: `object` 当弹窗关闭时，处理的一个 promise。有一个额外的 `关闭` 函数，可以被带有任何给定的值的关闭程序调用。

`confirm(可选)`

显示一个简单的带有一个取消和OK按钮的对话框弹窗。

如果用户点击OK按钮，就设置promise为true，如果用户点击取消按钮则为false。

参数	类型	详情
options	`object`	显示对话框弹窗选项的形式

```
{
  title: '', // String. 弹窗标题。
  subTitle: '', // String (可选)。弹窗的副标题。
  template: '', // String (可选)。放在弹窗body内的html模板。
  templateUrl: '', // String (可选)。放在弹窗body内的一个html模板的URL。
  cancelText: '', // String (默认: 'Cancel')。一个取消按钮的文字。
  cancelType: '', // String (默认: 'button-default')。取消按钮的类型。
  okText: '', // String (默认: 'OK')。OK按钮的文字。
  okType: '', // String (默认: 'button-positive')。OK按钮的类型。
}
```

- 返回: `object` 当关闭对话框时，处理的一个promise。当弹窗关闭时，处理的一个promise。有一个额外的 `关闭` 函数，可以被带有任何给定的值的关闭程序调用。

`prompt(可选)`

显示一个简单的提示弹窗，带有一个input， OK 按钮，和取消按钮。如果用户点击OK，就设置promise的值，如果用户点击取消，则值为未定义。

```
$ionicPopup.prompt({
  title: 'Password Check',
  template: 'Enter your secret password',
  inputType: 'password',
  inputPlaceholder: 'Your password'
}).then(function(res) {
  console.log('Your password is', res);
});
```

参数	类型	详情
options	`object`	显示的提示弹窗选项的形式

```
{
  title: '', // String. 弹窗的标题。
  subTitle: '', // String (可选)。弹窗的副标题。
  template: '', // String (可选)。放在弹窗body内的html模板。
  templateUrl: '', // String (可选)。放在弹窗body内的html模板的URL。
  inputType: // String (默认: 'text')。input的类型。
  inputPlaceholder: // String (默认: '')。input的 placeholder。
  cancelText: // String (默认: 'Cancel')。取消按钮的文字。
  cancelType: // String (默认: 'button-default')。取消按钮的类型。
  okText: // String (默认: 'OK')。OK按钮的文字。
  okType: // String (默认: 'button-positive')。OK按钮的类型。
}
```

- 返回: `object` 当关闭对话框时，处理的一个promise。当弹窗关闭时，处理的一个promise。有一个额外的`关闭`函数，可以被带有任何给定的值的关闭程序调用。

Loading（加载）

$ionicLoading

用一个覆盖层表示当前处于活动状态，来阻止用户的交互动作。

用法

```
angular.module('LoadingApp', ['ionic'])
.controller('LoadingCtrl', function($scope, $ionicLoading) {
  $scope.show = function() {
    $ionicLoading.show({
      template: 'Loading...'
    });
  };
  $scope.hide = function(){
    $ionicLoading.hide();
  };
});
```

方法

`show(opts)`

显示一个loading指示器。如果该指示器已经显示，它会设置给定选项，并保持指示器显示。

参数	类型	详情
opts	`object`	loading指示器的选项。

可用属性：

- `{string=}` `template` 指示器的html内容。
- `{string=}` `templateUrl` 一个加载html模板的url作为指示器的内容。
- `{boolean=}` `noBackdrop` 是否隐藏背景。默认情况下它会显示。
- `{number=}` `delay` 指示器延迟多少毫秒显示。默认为不延迟。
- `{number=}` `duration` 等待多少毫秒后自动隐藏指示器。默认情况下，指示器会一直显示，直到触发 `.hide()` 。

`hide()`

隐藏loading指示器，如果它已显示。

Platform（平台）

$ionicPlatform

一个angular抽象的 `ionic.Platform` 。

用来检测当前的平台，以及诸如在PhoneGap/Cordova中覆盖Android后退按钮。

方法

`onHardwareBackButton(callback)`

一些平台有硬件的后退按钮，因此可以用这种方法绑定到它。

参数	类型	详情
callback	`function`	当该事件发生时，触发回调函数。

`offHardwareBackButton(callback)`

移除一个后退按钮的监听事件。

参数	类型	详情
callback	`function`	最初绑定的监视器函数。

`registerBackButtonAction(callback, priority, [a`

注册一个硬件后退按钮动作。当点击按钮时，只有一个动作会执行，因此该方法决定了注册的后退按钮动作具有最高的优先级。

例如，如果一个动作表已经显示，后退按钮应该关闭这个动作表，但是它不应该还能返回一个页面视图或关闭一个打开的模型。

参数	类型	详情
callback	`function`	当点击返回按钮时触发，如果该监视器具有最高的优先级。
priority	`number`	仅最高优先级的会执行。
actionId(可选)	`*`	该id指定这个动作。默认：一个随机且唯一的id。

- 返回: `function` 一个被触发的函数，将会注销该后退按钮动作。

`ready([callback])`

一旦设备就绪，则触发一个回调函数，或如果该设备已经就绪，则立即调用。

参数	类型	详情
callback*(可选)*	`function=`	触发的函数。

- 返回: `promise` 当设备就绪后，就会解决一个 promise。

Gesture（手势）

$ionicGesture

一个angular服务展示ionic `ionic.EventController` 手势。

方法

`on(eventType, callback, $element)`

在一个元素上添加一个事件监听器。参加 `ionic.EventController` 。

参数	类型	详情
eventType	`string`	监听的手势事件。
callback	`function(e)`	当手势事件发生时触发的事件。
$element	`element`	angular元素监听的事件。

`off(eventType, callback, $element)`

在一个元素上移除一个手势事件监听器。参加 `ionic.EventController` 。

参数	类型	详情
eventType	`string`	移除监听的手势事件。
callback	`function(e)`	移除监听器。
$element	`element`	被监听事件的angular元素。

Backdrop（背景）

$ionicBackdrop

显示和隐藏UI上的背景，popups，loading，和其他覆盖层形式出现。

通常，多个UI组件需要一个背景，但是只有一个背景经常需要同时在DOM中。

因此，每个组件都需要显示背景，当它需要背景时，调用 `$ionicBackdrop.retain()` ，然后，当完成时，调用 `$ionicBackdrop.release()` 。

每次 `retain` 被调用时，背景会显示出来直到调用 `release` 。

比如，如果 `retain` 被调用三次，背景就会显示，知道 `release` 被调用三次。

用法

```
function MyController($scope, $ionicBackdrop, $timeout) {
  //一秒显示一个背景
  $scope.action = function() {
    $ionicBackdrop.retain();
    $timeout(function() {
      $ionicBackdrop.release();
    }, 1000);
  };
}
```

Methods

`retain()`

保留背景。

`release()`

释放背景。

Utility（工具）

ionic.Platform

方法

`ready(callback)`

设备就绪后触发一个回调函数，或如果设备已经就绪理解触发。该方法可以随处运行而无需通过任何附加方法封装。当一个app包含一个web视图（Cordova），当设备就绪后它将会触发回调函数。如果该app包含一个web浏览器，它会在 `window.load` 之后触发回调。

参数	类型	详情
callback	`function`	调用的函数

`device()`

返回当前设备（通过cordova提供）。

- 返回: `对象` 设备对象。

`isWebView()`

- 返回: `boolean` 验证我们是否附带web视图运行（比如Cordova）。

`isIPad()`

- 返回: `boolean` 是否在iPad上运行。

`isIOS()`

- 返回: `boolean` 是否在iOS上运行。

`isAndroid()`

- 返回: `boolean` 是否在Android上运行。

`isWindowsPhone()`

- 返回: `boolean` 是否在Windows手机上运行。

`platform()`

- 返回: `string` 当前平台的名字。

`version()`

- 返回: `string` 当前设备平台的版本。

`exitApp()`

退出app。

`showStatusBar(shouldShow)`

显示或隐藏设备状态栏（用Cordova）。

参数	类型	详情
shouldShow	`boolean`	是否显示状态栏。

`fullScreen([showFullScreen], [showStatusBar])`

设置app是否全屏（用Cordova）。

参数	类型	详情
showFullScreen*(可选)*	`boolean`	是否设置app全屏。默认为true。
showStatusBar*(可选)*	`boolean`	是否显示设备状态栏。默认为false。

属性

- **`boolean`** **`isReady`**

  设备是否就绪。

- **`boolean`** **`isFullScreen`**

  设备是否全屏。

- `Array(string)` `platforms`

  一个所有平台的数组。

- `string` `grade`

  当前平台是什么级别的。

ionic.DomUtil

方法

`requestAnimationFrame(callback)` (别号: ionic.requestAnimationFrame)

触发requestAnimationFrame，或一个polyfill如果不可用。

参数	类型	详情
callback	`function`	触发下一个框架时，调用该函数。

`animationFrameThrottle(callback)` (别号: ionic.animationFrameThrottle)

当给定一个回调函数时，如果在框架动画之间，被调用100次，添加 Throttle 将会使它只运行最后的100次调用。

参数	类型	详情
callback	`function`	一个函数会被requestAnimationFrame终止。

- 返回: `function` 一个函数会传递一个回调。回调传递接收的内容返回给被调用的函数。

`getPositionInParent(element)`

获取一个元素在容器内滚动的偏移。

参数	类型	详情
element	`DOMElement`	找到便宜的元素。

- 返回: `object` 一个位置对象具有如下属性：
  - `{number}` `left` 元素的左偏移。
  - `{number}` `top` 元素的上偏移。

`ready(callback)`

当DOM就绪后调用一个函数，或如果它已经就行则立即调用。

参数	类型	详情
callback	`function`	被掉用的函数。

`getTextBounds(textNode)`

获取一个矩形占用的textNode给定的边界。

参数	类型	详情
textNode	`DOMElement`	textNode查找的边界。

- 返回: `object` 一个对象占据边界的节点。属性：
    - `{number}` `left` textNode左侧位置。
    - `{number}` `right` textNode右侧位置。
    - `{number}` `top` textNode上位置。
    - `{number}` `bottom` textNode下位置。
    - `{number}` `width` textNode的宽度。
    - `{number}` `height` textNode的高度。

`getChildIndex(element, type)`

在给定的元素的指定类型内获取第一个子节点的索引。

参数	类型	详情
element	`DOMElement`	找到索引的元素。
type	`string`	与子元素对应的节点名称。

- 返回: `number` 索引，或-1，匹配类型的子节点名称。

`getParentWithClass(element, className)`

参数	类型	详情
element	`DOMElement`	
className	`string`	

- 返回: `DOMElement` 匹配最近的父元素类名，或为空。

`getParentWithClass(element, className)`

参数	类型	详情
element	`DOMElement`	
className	`string`	

- 返回: `DOMElement` 匹配最近的父元素或自身，或为空。

`rectContains(x, y, x1, y1, x2, y2)`

参数	类型	详情
x	`number`	
y	`number`	
x1	`number`	
y1	`number`	
x2	`number`	
y2	`number`	

- 返回: `boolean` 由{x1,y1,x2,y2}定义的矩形内部是否与{x,y}匹配。

ionic.EventController

方法

`trigger(eventType, data, [bubbles], [cancelable` (别号: ionic.trigger)

参数	类型	详情
eventType	`string`	触发的事件。
data	`object`	事件的数据。提示： 通过 `{target: targetElement}` 传递。
bubbles*(可选)*	`boolean`	事件是否在DOM中冒泡。
cancelable*(可选)*	`boolean`	事件是否能被取消。

`on(type, callback, element)` (别号: ionic.on)

监听一个元素上的事件。

参数	类型	详情
type	`string`	监听的事件。
callback	`function`	触发监听器。
element	`DOMElement`	监听该事件的元素。

`off(type, callback, element)` (别号: ionic.off)

移除一个事件的监听器。

参数	类型	详情
type	`string`	
callback	`function`	
element	`DOMElement`	

`onGesture(eventType, callback, element)` (别号: ionic.onGesture)

在一个元素上添加一个手势事件监听器。

可用的事件类型（来自hammer.js）：

`hold`, `tap`, `doubletap`, `drag`, `dragstart`, `dragend`, `dragup`, `dragdown`, `dragleft`, `dragright`, `swipe`, `swipeup`, `swipedown`, `swipeleft`, `swiperight`, `transform`, `transformstart`, `transformend`, `rotate`, `pinch`, `pinchin`, `pinchout`, `touch`, `release`

参数	类型	详情
eventType	`string`	监听的手势事件。
callback	`function(e)`	当手势发生时，触发的函数。
element	`DOMElement`	监听事件的angular元素。

`offGesture(eventType, callback, element)` (别号: ionic.offGesture)

移除一个元素上的事件监听器。

参数	类型	详情
eventType	`string`	手势事件
callback	`function(e)`	之前添加的监听器。
element	`DOMElement`	被监听的元素。

Keyboard（键盘）

keyboard

在Android 和 iOS中, Ionic会试图阻止键盘的模糊输入以及聚焦元素，当在视图中滚动出现的时候。为了这项工作，任何可以获取焦点的元素必须在一个滚动视图或一个类似于带有滚动视图的Content指令内。

在获取焦点时，它会试图阻止原生的滚动溢出，这可能会导致布局问题，比如将header挤到上面，并超出视野。

键盘修复可以和Ionic键盘插件最好的协同工作，尽管没有它，Ionic键盘插件也会执行良好。然而，如果你使用Cordova的话，就没有理由用该插件。

当键盘显示的时候隐藏

当键盘被打开的时候，要隐藏一个元素，添加 `hide-on-keyboard-open` 类。

```
<div class="hide-on-keyboard-open">
  <div id="google-map"></div>
</div>
```

插件用法

使用该插件的用法可以参考 https://github.com/driftyco/ionic-plugins-keyboard 。

Android平台注意事项

- 如果你的app全屏运行，即 `config.xml` 文件内有 `<preference name="Fullscreen" value="true" />` ，你需要手动设置 `ionic.Platform.isFullScreen = true` 。
- 你可以配置web视图的行为，通过设置android:windowSoftInputMode或 `adjustPan` 来显示键盘，在你app中 `AndroidManifest.xml` 的 `adjustResize` 或 `adjustNothing` 行为。 `adjustResize` 为Ionic推荐设置，但是如果处于一些原因你使用了 `adjustPan` ，那么你需要设置 `ionic.Platform.isFullScreen = true` 。

  ```
  &lt;activity android:windowSoftInputMode="adjustResize"&gt;
  ```

iOS平台注意事项

- 如果在input获取焦点时，你app的内容（包含header）被挤到上面或视图以外，就需要尝试设置 `cordova.plugins.Keyboard.disableScroll(true)` 。这并不会在Ionic滚动视图中禁用滚动，相反，它会禁用原生的滚动溢出，当自动发生input获取焦点在键盘之后时。

keyboard-attach

keyboard-attach是一个属性指令，在键盘显示时，它会导致一个元素悬浮在键盘上方。目前仅支持ion-footer-bar指令。

注意

- 该指令依赖Ionic键盘插件。
- Android设备未全屏模式中，即，在你的 `config.xml` 文件里设置了 `<preference name="Fullscreen" value="true" />` ，该指令是不必要的，因为她默认的行为。
- 在iOS中，在你的footer中有一个input，你需要设置 `cordova.plugins.Keyboard.disableScroll(true)` 。

用法

```
<ion-footer-bar align-title="left" keyboard-attach class="bar-assertive">
  <h1 class="title">标题!</h1>
</ion-footer-bar>
```